

[illegible]

MM	MM	000000	NN	NN	IIIIII	TTTTTTTTTT	000000	RRRRRRRR	
MM	MM	000000	NN	NN	IIIIII	TTTTTTTTTT	000000	RRRRRRRR	
MMM	MMM	00	NN	NN	II	TT	00	RR	RR
MMM	MMM	00	NN	NN	II	TT	00	RR	RR
MM	MM	00	NNNN	NN	II	TT	00	RR	RR
MM	MM	00	NNNN	NN	II	TT	00	RR	RR
MM	MM	00	NN	NN	II	TT	00	RRRRRRRR	
MM	MM	00	NN	NN	II	TT	00	RRRRRRRR	
MM	MM	00	NN	NNNN	II	TT	00	RR	RR
MM	MM	00	NN	NNNN	II	TT	00	RR	RR
MM	MM	00	NN	NN	II	TT	00	RR	RR
MM	MM	00	NN	NN	II	TT	00	RR	RR
MM	MM	000000	NN	NN	IIIIII	TT	000000	RR	RR
MM	MM	000000	NN	NN	IIIIII	TT	000000	RR	RR

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	211	DECLARATIONS
(13)	848	CALC_LEN - Calculate class record lengths
(15)	1076	MOVE_CLASS_QUALS - Move Class Qualifier Values
(16)	1214	FETCH - Collect Data into Buffer
(17)	1314	CLASS_COLLECT - Collect & Transform Data
(20)	1594	TRANSFORMS - Perform Data Transformations
(21)	1681	COMBINE_MODES - Combine Modes for all CPUs
(22)	1737	QUAD_LT_QUAD - Compare Two Quadwords
(23)	1798	QUAD_EQ_0 - Compare Quadword = 0
(24)	1857	MPCHECK - Check system for MP capability
(25)	1916	WRITE_PROC_RECORDS - Write PROCESSES class records
(28)	2068	CVT_TO_DELTA - Convert Seconds to Delta
(29)	2115	COMPUTE_BOOTTIME - Compute System Time of Boot
(30)	2178	CLUS_NET_INFO - Get Cluster & Net Info
(31)	2329	ADV_ROM_ITEM - Advance to next display item for homog
(32)	2406	COLLECTION - Collect into CURRENT Buffer
(35)	2583	FILL_HETERO_STATS - Fill the STATS Buffer
(37)	2771	FILL_PCSTATS_BUFF - Fill PCSTATS Buffer from STATS Buffer
(38)	2831	COMPUTE_STATS - Statistical Computations on STATS
(39)	2949	UPD_PC_MIN_MAX - Update Percent Min/Max Buffers
(41)	3017	DISPLAY_INIT - Init for Display Output
(42)	3156	SUMMARY_INIT - Init for Summary Output
(44)	3294	FILL_DISP_BUFF - Fill Display Buffer
(56)	4094	FDB_SYS_TOP - Process TOPs for SYSTEM class
(60)	4402	FILL_TOP - Fill Display Buffer for TOP PROCESSES
(66)	4732	SUMMARY_TOP - Set up Summary for TOP
(69)	4872	DISPLAY_PROCS - Put PROCESSES Display Output to Screen
(78)	5244	DISPLAY_TOP - Put PROCESSES/TOP Display Output to Screen
(79)	5345	DISPLAY_HOMOG - Put Homog Class Display Output to Screen
(86)	5561	FILL_HOMOG_SCREEN - Fill a Screen with Homog Class Output
(88)	5758	DISP_HOM_NAMES - Display Names for Homog Class
(90)	5878	DISPLAY_PUT - Put Display Output to Screen
(91)	5967	PUT_TO_SCREEN - Translate escape seqs and issue PUT_SCREEN
(92)	6125	SELECT_REV_LEVS - Select Revision Levels
(93)	6305	ESTAB_CTRLZ - Establish CTRL-C,Z Handlers
(94)	6386	ESTAB_CTRLW - Establish CTRL-W Handler
(95)	6461	MON_ERR - Log MONITOR Error
(96)	6545	SIGNALED_ERR - Log Signaled Error
(97)	6620	SIGNAL_MON_ERR - Signal MONITOR Error
(98)	6665	LINK_MON_ERR - Link MONITOR Error
(99)	6728	FREE_MEM - Free Virtual Memory
(100)	6856	DISK_DISPNAME - DISK Class display name subroutine
(103)	7007	SCS_DISPNAME - SCS Class display name subroutine

```
0000 1      .TITLE  MONITOR - VAX/VMS Performance Monitor Utility
0000 2      .IDENT  'V04-000'
0000 3
0000 4
0000 5      *****
0000 6      *
0000 7      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      *  ALL RIGHTS RESERVED.
0000 10     *
0000 11     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     *  TRANSFERRED.
0000 17     *
0000 18     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     *  CORPORATION.
0000 21     *
0000 22     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     *
0000 25     *
0000 26     *****
0000 27
0000 28     ++
0000 29     FACILITY:  VAX/VMS MONITOR Utility
0000 30
0000 31     ABSTRACT:
0000 32
0000 33         This module is a collection of routines which are called by
0000 34         the MONMAIN, REQUEST, and COLLEVT PL/I routines to do
0000 35         various tasks, including data collection, terminal I/O
0000 36         (through the use of the Screen Package) and maintenance
0000 37         of statistics for screen displays.
0000 38
0000 39     ENVIRONMENT:
0000 40
0000 41         Unprivileged user mode,
0000 42         except for certain collection routines which
0000 43         run in EXEC or KERNEL mode to access system
0000 44         data bases.
0000 45
0000 46     AUTHOR: Thomas L. Cafarella, April, 1981
0000 47
0000 48     MODIFIED BY:
0000 49
0000 50         V03-027 TLC1090      Thomas L. Cafarella      02-Aug-1984      15:00
0000 51         Correct ACCVIOs in SYSTEM and PROCESSES classes.
0000 52
0000 53         V03-026 TLC1087      Thomas L. Cafarella      25-Jul-1984      15:00
0000 54         Default to /ALL when summarizing.
0000 55
0000 56         V03-025 TLC1086      Thomas L. Cafarella      24-Jul-1984      14:00
0000 57         Make top summary work for SYSTEM class.
```


0000	58	:				
0000	59	:	V03-024	TLC1085	Thomas L. Cafarella	22-Jul-1984 14:00
0000	60	:		Calculate scale values for Free and Modified List bar graphs.		
0000	61	:				
0000	62	:	V03-024	TLC1083	Thomas L. Cafarella	20-Jul-1984 11:00
0000	63	:		If counter value decreases, use 0 for delta.		
0000	64	:				
0000	65	:	V03-023	TLC1081	Thomas L. Cafarella	18-Jul-1984 11:00
0000	66	:		Correct use of R6 in PROCESSES /TOP display.		
0000	67	:				
0000	68	:	V03-022	TLC1078	Thomas L. Cafarella	11-Jul-1984 11:00
0000	69	:		Use SCSNODE node name if present before using SYSSNODE.		
0000	70	:				
0000	71	:	V03-021	TLC1072	Thomas L. Cafarella	17-Apr-1984 11:00
0000	72	:		Add volume name to DISK display.		
0000	73	:				
0000	74	:	V03-020	PRS1019	Paul R. Senn	11-Apr-1984 16:00
0000	75	:		Fix /SUMMARY for SYSTEM class.		
0000	76	:				
0000	77	:	V03-020	PRS1018	Paul R. Senn	11-Apr-1984 9:00
0000	78	:		Display CPU busy instead of CPU idle in SYSTEM class.		
0000	79	:				
0000	80	:	V03-020	TLC1066	Thomas L. Cafarella	01-Apr-1984 11:00
0000	81	:		Add SYSTEM class.		
0000	82	:				
0000	83	:	V03-020	TLC1063	Thomas L. Cafarella	3-Apr-1984 13:00
0000	84	:		Add check to ensure that a counter which is re-init'd to		
0000	85	:		zero will not cause an **** to be displayed.		
0000	86	:				
0000	87	:	V03-020	PRS1015	Paul R. Senn	3-Apr-1984 15:00
0000	88	:		add shared error message capability		
0000	89	:				
0000	90	:	V03-020	TLC1062	Thomas L. Cafarella	31-Mar-1984 23:00
0000	91	:		Fix bug causing summary averages to be displayed as zeroes		
0000	92	:		for homogeneous classes.		
0000	93	:				
0000	94	:	V03-019	TLC1061	Thomas L. Cafarella	18-Mar-1984 11:00
0000	95	:		Identify dual-path disks by allocation class.		
0000	96	:				
0000	97	:	V03-019	TLC1060	Thomas L. Cafarella	12-Mar-1984 11:00
0000	98	:		Make multi-file summary work for homogeneous classes.		
0000	99	:				
0000	100	:	V03-018	PRS1008	Paul R. Senn	17-FEB-1984 14:00
0000	101	:		Move GET_BUFFERS and associated subroutines into separate		
0000	102	:		module.		
0000	103	:				
0000	104	:	V03-018	PRS1006	Paul R. Senn	17-FEB-1984 14:00
0000	105	:		Add support for "computed" items		
0000	106	:				
0000	107	:	V03-018	TLC1052	Thomas L. Cafarella	17-Feb-1984 11:00
0000	108	:		Add multi-file summary capability.		
0000	109	:				
0000	110	:	V03-017	PRS1005	Paul R. Senn	13-JAN-1984 10:00
0000	111	:		Allow flexible spacing between screen items		
0000	112	:				
0000	113	:	V03-016	TLC1051	Thomas L. Cafarella	11-Jan-1984 11:00
0000	114	:		Add consecutive number to class header record.		

0000	115	:	V03-016	PRS1002	Paul R. Senn	29-Dec-1983	16:00
0000	116	:		Add YES and NO global equated symbols, which are no longer			
0000	117	:		defined globally by MONMAIN,			
0000	118	:					
0000	119	:					
0000	120	:	V03-016	PRS1000	Paul R. Senn	15-Dec-1983	16:00
0000	121	:		For cases where one display event may involve multiple			
0000	122	:		screens of data (such as PROCESSES and Homogeneous			
0000	123	:		classes), make the wait between screens = VIEWING_TIME,			
0000	124	:		instead of a constant of 2 seconds.			
0000	125	:					
0000	126	:	V03-015	TLC1050	Thomas L. Cafarella	06-Dec-1983	11:00
0000	127	:		Change directory information in DLOCK class.			
0000	128	:					
0000	129	:	V03-015	TLC1049	Thomas L. Cafarella	10-Oct-1983	15:00
0000	130	:		Position TOP bar display properly.			
0000	131	:					
0000	132	:	V03-014	TLC1048	Thomas L. Cafarella	11-Sep-1983	12:00
0000	133	:		Remove UIC from PROCESSES displays.			
0000	134	:					
0000	135	:	V03-013	TLC1047	Thomas L. Cafarella	09-Sep-1983	10:00
0000	136	:		De-establish CTRL/W handler to get back AST quota.			
0000	137	:					
0000	138	:	V03-012	TLC1043	Thomas L. Cafarella	30-Jul-1983	13:00
0000	139	:		Eliminate special characters from node name.			
0000	140	:					
0000	141	:	V03-012	SPC0004	Stephen P. Carney	24-Jun-1983	16:00
0000	142	:		Add SCS display subroutine for new SCS Class. Add FAO and			
0000	143	:		ASCII string for SCS class.			
0000	144	:					
0000	145	:	V03-011	TLC1042	Thomas L. Cafarella	19-Jun-1983	15:00
0000	146	:		Add /ITEM qualifier for homogeneous classes.			
0000	147	:					
0000	148	:	V03-011	TLC1039	Thomas L. Cafarella	15-Jun-1983	15:00
0000	149	:		Add DECnet node name to heading.			
0000	150	:					
0000	151	:	V03-011	TLC1036	Thomas L. Cafarella	10-Jun-1983	15:00
0000	152	:		Properly recognize Revision Level 0.			
0000	153	:					
0000	154	:	V03-010	TLC1035	Thomas L. Cafarella	06-Jun-1983	15:00
0000	155	:		Add homogeneous class type and DISK class.			
0000	156	:					
0000	157	:	V03-009	TLC1030	Thomas L. Cafarella	25-Apr-1983	10:00
0000	158	:		Initialize MIN and MAX buffers.			
0000	159	:					
0000	160	:	V03-009	TLC1029	Thomas L. Cafarella	21-Apr-1983	10:00
0000	161	:		Correctly calculate "Interrupt Stack" string.			
0000	162	:					
0000	163	:	V03-008	TLC1028	Thomas L. Cafarella	14-Apr-1983	16:00
0000	164	:		Add interactive user interface.			
0000	165	:					
0000	166	:	V03-008	TLC1027	Thomas L. Cafarella	14-Apr-1983	16:00
0000	167	:		Enhance file compatibility features.			
0000	168	:					
0000	169	:	V03-008	SPC0001	Stephen P. Carney	25-Mar-1983	15:00
0000	170	:		Add RWxxx and MUTEX states in place of MWAIT state.			
0000	171	:					

0000	172	:	V03-007	TLC1024	Thomas L. Cafarella	1-Mar-1983	11:00
0000	173	:		Convert an IPID to an EPID before placing it on the FAO			
0000	174	:		stack (for MONITOR PROCESSES display).			
0000	175	:					
0000	176	:	V03-006	TLC1021	Thomas L. Cafarella	07-Jul-1982	16:00
0000	177	:		Change \$\$\$DEF symbols to GLOBAL since they are no longer			
0000	178	:		accessible at link time.			
0000	179	:					
0000	180	:	V03-005	TLC1018	Thomas L. Cafarella	12-Apr-1982	16:00
0000	181	:		Collect all data at KERNEL mode instead of EXEC.			
0000	182	:					
0000	183	:	V03-004	TLC1016	Thomas L. Cafarella	02-Apr-1982	16:00
0000	184	:		Replace references to EXESGQ_SYSTIME with \$GETTIM calls.			
0000	185	:					
0000	186	:	V03-004	TLC1015	Thomas L. Cafarella	01-Apr-1982	16:00
0000	187	:		Change .PSECT options in order to group image sections.			
0000	188	:					
0000	189	:	V03-004	TLC1014	Thomas L. Cafarella	01-Apr-1982	13:00
0000	190	:		Correct attached processor time reporting for MODES class.			
0000	191	:					
0000	192	:	V03-004	TLC1012	Thomas L. Cafarella	30-Mar-1982	13:00
0000	193	:		Display user's comment string on screen line 5.			
0000	194	:					
0000	195	:	V03-003	TLC1008	Thomas L. Cafarella	28-Mar-1981	21:00
0000	196	:		Fix to display first and last PROCESSES records on playback.			
0000	197	:					
0000	198	:	V03-002	TLC1002	Thomas L. Cafarella	20-Mar-1981	13:00
0000	199	:		Change PROCESSES display from scroll-style to page-style to			
0000	200	:		make it terminal-independent.			
0000	201	:					
0000	202	:		Widen working set field of PROCESSES display.			
0000	203	:					
0000	204	:		Reset DEC_CRT advanced video options at exit.			
0000	205	:					
0000	206	:	V03-001	TLC1001	Thomas L. Cafarella	16-Mar-1981	13:00
0000	207	:		Add CTRL-W screen refresh support.			
0000	208	:					
0000	209	:--					


```
0000 211      .SBTTL  DECLARATIONS
0000 212      .PSECT  MONDATA,QUAD,NOEXE
0000 213      :
0000 214      : INCLUDE FILES:
0000 215      :
0000 216      :
0000 217      $DCDEF      : define device class codes
0000 218      $DIBDEF     : define device information block
0000 219      $DSCDEF     : Descriptor Definitions
0000 220      $IODEF      : insert I/O function codes
0000 221      $IPLDEF     : define interrupt levels
0000 222      $JPIDEF     : define GETJPI items
0000 223      $PCBDEF      : process control block
0000 224      $PHDDEF     : process header definitions
0000 225      $PRDEF      : define processor registers
0000 226      $PSLDEF     : define PSL fields
0000 227      $RPBDEF     : define Restart Parameter Block
0000 228      $RSNDEF     : define resource wait codes
0000 229      $SCRDEF     : SCRPKG definitions
0000 230      $SSDEF      : define status codes
0000 231      $STATEDEF   : define process state codes
0000 232      $STSDEF     : define status fields
0000 233      $SYIDEF     : define GETSYI item identifiers
0000 234      $SHRDEF     : define shared error codes
0000 235      $TTDEF      : define terinal functions
0000 236      :
0000 237      $CDBDEF     : Define Class Descriptor Block
0000 238      $CDXDEF     : Define CDB Extension
0000 239      $CHDDEF     : Define Change Descriptor
0000 240      $IDBDEF     : define item descriptor block offsets
0000 241      $MRBDEF     : Define Monitor Request Block
0000 242      $MBPDEF     : Define Monitor Buffer Pointers
0000 243      $MCADEF     : Define Monitor Communication Area
0000 244      $MONDEF     : Monitor Recording File Definitions
0000 245      $SCBDEF     : Define STATS Control Block
0000 246      $MFSDEF     : Define Multi-File Summary Block
0000 247      $TM4DEF     : Define temporary storage
0000 248      :
0000 249      :
0000 250      : MACROS:
0000 251      :
0000 252      :
0000 253      :
0000 254      : Local Macro Definitions
0000 255      :
0000 256      :
0000 257      :
0000 258      : ALLOC Macro - Dynamically allocate space on the stack.
0000 259      :
0000 260      :
0000 261      .MACRO  ALLOC  LENGTH,RSLDESC,RSLBUF
0000 262      SUBL    #<LENGTH+3>&<^C3>,SP
0000 263      .IF     NB,RSLBUF
0000 264      MOVL    SP,RSLBUF
0000 265      .ENDC
0000 266      PUSHL   SP
0000 267      PUSHL   #LENGTH
```

MONITOR
V04-000

- VAX/VMS Performance Monitor Utility L 12
DECLARATIONS

16-SEP-1984 01:59:24
5-SEP-1984 02:01:24

VAX/VMS Macro V04-00
[MONITOR.SRC]MONITOR.MAR;1

Page 6
(2)

0000	268	MOVL	SP,RSLDESC
0000	269	.ENDM	ALLOC
0000	270		

MMNN

NNN


```
0000 272 :  
0000 273 : EQUATED SYMBOLS:  
0000 274 :  
0000 275 :  
0000 276 :  
00000001 0000 277 YES == 1  
00000000 0000 278 NO == 0  
00000000 0000 279 DEF_TO_CUR == 0  
00000001 0000 280 CUR_TO_ACT == 1  
00000002 0000 281 ACT_TO_CUR == 2  
00000003 0000 282 ALL_TO_ACT == 3  
0000 283 :  
00000002 0000 284 COLL_BUFS == 2  
00000004 0000 285 REG_BUFS == 4  
00000004 0000 286 PC_BUFS == 4  
0000 287 :  
7FFF7FFF 0000 288 LARGE_NO == ^X7FFF7FFF  
0000 289 :  
0000000D 0000 290 CR = 13  
0000001B 0000 291 ESC = 27  
0000000A 0000 292 LF = 10  
0000000F 0000 293 SI = 15  
00000002 0000 294 BS_SECS = 2  
00000028 0000 295 MAXBARS == 40  
0000001A 0000 296 MAXBARS_SYS == 26  
0000002A 0000 297 DEF_BAR = ^A/*/  
00000061 0000 298 VID_BAR = ^A/a/  
000000C8 0000 299 MAX55HEIGHT = 200  
00000032 0000 300 STARTPOS = 50  
00000026 0000 301 START_XPOS = 34+4  
00000018 0000 302 VTHEIGHT == 24  
00000050 0000 303 VTWIDTH == 80  
00000008 0000 304 FIRST_DATA_LINE == 8  
00000016 0000 305 LAST_DATA_LINE == 22  
0000000F 0000 306 VTDATALINES == LAST_DATA_LINE - FIRST_DATA_LINE + 1  
0000 307 :  
0000004A 0000 308 VT55CWIDTH == 74  
000000EC 0000 309 VT55HEIGHT = 236  
00000200 0000 310 VT55WIDTH = 512  
00000005 0000 311 NAME_COL_TAB == 5  
00000002 0000 312 NAME_COL_BAR == 2  
00000001 0000 313 NAME_COL_MFSUM == 1  
0000004B 0000 314 PROC_LINE = 75  
00000000 0000 315 PUTS_REGSET = 0  
00000000 0000 316 PUTS_ALTSET = 0  
000000C8 0000 317 MAXELTS == 200  
00000190 0000 318 MAXELTS_MFS == MAXELTS*2  
0000001B 0000 319 MAX_ELIDLEN == 27  
0000000F 0000 320 MAX_HOM_ITEMS == 15  
00000200 0000 321 SCRDISC_SIZE = 512  
00001900 0000 322 FAOSTK_SIZE = MAXELTS*8*4  
000006A4 0000 323 OUTDSC_SIZE = 1700  
000005DC 0000 324 FAOCTR_SIZE == 1500  
0000001A 0000 325 PUTMSG_SIZE = 26  
00007D00 0000 326 MAX_REC_SIZE == 32000  
00000000 0000 327 SYS_FAC_NO = 0  
00000008 0000 328 TAB_LWORDS = 8
```

: Codes for MOVE_CLASS_QUALS routine:
: Yep
: Nope
: Move default values to current values
: Move current values to active values
: Move active values to current values
: Move ALL statistic value to active

: Number of collection buffers
: Number of "regular stats" buffers
: Number of "percent stats" buffers

: Very large number (integer or float)

: carriage return
: escape character
: line feed
: shift in (selects G0 char set on VT100)
: seconds between screens (of a mult scr)
: number of bar chars in horiz. graph
: same as above for SYSTEM display
: default bar character (for hardcopy)
: video terminal bar character
: max height of vertical bars (VT55)
: start of bargraph position
: starting x position for VT55 bar
: number of rows on screen
: number of columns on screen
: line no of 1st data line on screen
: line no of last data line on screen
: number of lines of actual data on screen
: number of chars. on x axis
: screen height in points
: screen width in points
: starting col no. for names -- tabular
: starting col no. for names -- bar graph
: starting col no. for names -- m.f. summ.
: width of a PROCESSES display line
: value for reg graphics set to PUT_SCREEN
: value for alt graphics set to PUT_SCREEN
: max no. of elements for a homog class
: max no. of homog elts in a m.f. summary
: max length of an element ID for homog
: max no. of items in a homog class
: screen package buffer size
: size of FAOL parameter stack for data disp
: size of FAO output buffer for displays
: size in bytes of FAO control string
: size in longwords of \$PUTMSG message arg v
: maximum record size for PLAYBACK and RECOR
: system facility code
: Define no. of FAOSTK longwords ...

```
0000 329 ; ... for tabular display. Used by
0000 330 ; ... FILL_HOMOG_SCREEN routine.
0000 331
00000003 0000 332 BAR_LWORDS = 3 ; Define no. of FAOSTK longwords ...
0000 333 ; ... for bar-graph display. Used by
0000 334 ; ... FILL_HOMOG_SCREEN routine.
0000 335
0000 336 ;
0000 337 ; SYS$OUTPUT Device types for Monitor
0000 338 ; Loaded into SYSOUT_TYPE
0000 339 ;
00000000 0000 340 DEC_CRT = 0 ; includes VT100-compatible devices
00000001 0000 341 VT5X = 1 ; VT5x series (VT52 and VT55)
00000002 0000 342 HARDCOPY = 2 ; hardcopy terminal and disk file
00000003 0000 343 OTHER_VID = 3 ; other video types
0000 344
0000 345 ;
0000 346 ; Monitor status codes which use shared error codes.
0000 347 ;
0000 348 ;
0000 349 ;
0000 350 ; **** NOTE **** The MONITOR facility number is defined here and in
0000 351 ; MONMSG.MSG. Any change needs to be made in BOTH places.
0000 352 ;
0000 353 ;
000000CE 0000 354 FACNO = 206 ; local symbol for monitor facility #
0000 355
00CE109A 0000 356 MNR$_OPENIN == <SHR$_OPENIN+ST$K_ERROR>!<FACNO@16> ;open-input-file error
```



```
0000 358 :  
0000 359 : OWN STORAGE:  
0000 360 :  
0000 361 :  
00000001 0000 362 NAME_COL:: .BLKB 1 ; column number of name string  
00000019 0001 363 BAR_SIZE:: .LONG 25 ; width of bar graph in points (VT55)  
00000006 0005 364 BAR_CHAR:: .BLKB 1 ; bar graph character  
0000000A 0006 365 BPU: .BLKF 1 ; no of bar chars per unit of output value  
0000000E 000A 366 GMIN: .BLKL 1 ; min value which bar graph can represent  
00000000 000E 367 CURGRAPH:: .LONG 0 ; currently enabled VT55 graph  
00000000 0012 368 CURXPOS: .LONG 0 ; current position for GRAPH6  
00000000 0016 369 PROCS_PER_REC: .LONG 0 ; number of processes which can fit into a .  
00000000 001A 370 ; ... single PLAYBACK or RECORD file class r  
00000000 001A 371 PROC_WRI_BUF: .LONG 0,0 ; PROCESSES write buffer descriptor  
00000023 0022 372 TOP_PROCS: .BLKB 1 ; number of top processes to display  
0000002B 0023 373 TOP_TIME: .BLKQ 1 ; collection time of most recent TOP  
0000002F 002B 374 ; ... display (in system time units)  
0000002F 002B 375 TOP_TICKS: .BLKL 1 ; number of clock ticks (10ms units)  
0000002F 002F 376 ; ... covered by most recent TOP display  
00000037 002F 377  
00000037 002F 378 S_TOP_TIME: .BLKQ 1 ; similar to above, but for SYSTEM  
0000003B 0037 379 ; ... class  
0000003B 0037 380 S_TOP_TICKS: .BLKL 1 ; ...  
0000003B 003B 381 ; ...  
0000007B 003B 382 SYS_TOP_VEC:: .BLKL 16 ; vector of ptrs to SYSTEM TOP arrays  
00000000 007B 383 SYS_DATA_ADDR:: .LONG 0 ; addr of the SYSTEM TOP arrays  
00000000 007F 384 SYS_DATA_LEN:: .LONG 0 ; len of the SYSTEM TOP arrays  
00000085 0083 385  
00000085 0083 386 SYS_BOX_STR_LEN:: .BLKW 1 ; len of SYSTEM box string  
00000089 0085 387 SYS_BOX_STR_ADDR:: .BLKL 1 ; ... and its address  
00000000 0089 388 NEWXPOS: .LONG 0 ; next point to plot for GRAPH6  
00000091 008D 389 PTS_STAT: .BLKL 1 ; save area for SCRPKG return stati  
00000000 0091 390  
00000000 0091 391 CTRLW_MASK: .LONG 0 ; mask required by QIO for out-of-band char  
00800000 0095 392 .LONG ^X00800000 ; bit 23 representing CTRL-W  
00000000 0099 393  
00000000 0099 394 CTRLZ_MASK: .LONG 0 ; mask required by QIO for out-of-band char  
04000000 009D 395 .LONG ^X04000000 ; bit 26 representing CTRL-Z  
00000000 00A1 396  
00000000 00A1 397 CTRLCZ_CHAN:: .LONG 0 ; channel no. for CTRL-C and -Z  
00000000 00A5 398 CTRLW_CHAN:: .LONG 0 ; channel no. for CTRL-W  
00000000 00A9 399  
FECED300 00A9 400 BET_SCREEN: .LONG -10*1000*1000*BS_SECS ; delta time between  
FFFFFFFF 00AD 401 .LONG -1 ; ... screens for single display event  
00000000 00B1 402  
4F 43 24 53 59 53 000000B9'010E0000' 00B1 403 SYSCMD_DESC: .ASCID \SYS$COMMAND\ ; User command terminal  
44 4E 41 4D 4D 00BF  
4F 4E 24 53 59 53 000000CC'010E0000' 00C4 404 SYSNOD_NAM: .ASCID \SYS$NODE\ ; DECnet node logical name  
45 44 00D2  
00000000 00D4 405  
00000000 00D4 406 VT55XINCR:: .LONG 0 ; incr to next bar of graph  
00000000 00D8 407  
000000E0 00D8 408 CB_ADDRS:: .BLKQ 1 ; holds coll buffer addrs in GET_BUFFERS  
000000E2 00E0 409 ITEM_TYPE: .BLKW 1 ; holds IDB item type code in FILE_DISP_BUFF  
000000E4 00E2 410 HOMOG_TYPE: .BLKW 1 ; holds same as above in COMPUTE_STATS  
000000E8 00E4 411  
000000E8 00E4 412 PREV_PD: .BLKL 1 ; no. of processes displayed in previous int
```

```

00E8 413
00E8 414 PROMPT_STR::
00E8 415 ; string and descriptor for
00E8 416 ; ... subcommand prompt string
0000000B' 00E8 416 .LONG 20$-10$
000000F0' 00EC 417 .LONG 10$
0A 0D 00F0 418 10$: .BYTE CR,LF
20 3E 52 4F 54 49 4E 4F 4D 00F2 419 .ASCII \MONITOR> \
00FB 420 20$:
00FB 421
00FB 422 DYN_STRING:: ; dynamic string descriptor for
00FB 423 ; ... use in MONMAIN.PLI
0000 00FB 424 .WORD 0 ; called routine will fill in length
0E 00FD 425 .BYTE DSC$K_DTYPE_T ; string descriptor type
02 00FE 426 .BYTE DSC$K_CLASS_D ; dynamic class
00000000 00FF 427 .LONG 0 ; called routine will fill in address
0103 428

```



```

0103 430 :
0103 431 ; FAO-related buffers required for /DISPLAY
0103 432 :
0103 433 :
00001A03 0103 434 FAOSTK:: .BLKB FAOSTK_SIZE ; DISPLAY buffer containing data for input t
1A03 435
1A03 436 OUTDSC:: ; FAO output buffer descriptor for data disp
000006A4 1A03 437 .LONG OUTDSC_SIZE
00001A0B' 1A07 438 .LONG 10$
000020AF 1A0B 439 10$: .BLKB OUTDSC_SIZE
20AF 440
20AF 441 SCRDCS:: ; Screen Package buffer descriptor
00000200 20AF 442 .LONG SCRDCS_SIZE
000020B7' 20B3 443 .LONG 10$
000022B7 20B7 444 10$: .BLKB SCRDCS_SIZE

```

```
22B7 446 :  
22B7 447 : Control and name strings for screen output.  
22B7 448 :  
22B7 449 : These strings contain embedded escape sequences. Before the strings  
22B7 450 : are sent to the Screen Package for output, the escape sequences are  
22B7 451 : interpreted (in the PUT TO SCREEN routine) and translated to general-  
22B7 452 : case screen package calls. The sequences embedded below are generally  
22B7 453 : equivalent to VT52 escape sequences, with the following exceptions:  
22B7 454 :  
22B7 455 : 1) Cursor addresses for both row and column are in the  
22B7 456 : form acceptable to the Screen Package -- i.e., the top  
22B7 457 : row is 1 and the left-hand column is 1.  
22B7 458 :  
22B7 459 : 2) ESC B means BOLD all text until an UNDO sequence.  
22B7 460 :  
22B7 461 : 3) ESC L means UNDERLINE all text until an UNDO sequence.  
22B7 462 :  
22B7 463 : 4) ESC R means REVERSE VIDEO all text until an UNDO sequence.  
22B7 464 :  
22B7 465 : 5) ESC U means UNDO all DEC_CRT advanced video attributes selected  
22B7 466 : (i.e., BOLD, UNDERLINE, REVERSE, BLINK)  
22B7 467 :  
22B7 468 :  
22B7 469 CLRVT55::  
08' 22B7 470 .BYTE 10$-5$  
22B8 471 5$:  
31 1B 22B8 472 .BYTE ESC,^A/1/ ; enter graphics mode  
20 41 22BA 473 .BYTE ^A/A/,^X20 ; turn off graphs  
30 49 22BC 474 .BYTE ^A/I/,^X30 ; turn off lines, cursors, etc.  
32 1B 22BE 475 .BYTE ESC,^A/2/ ; back to alphanumeric mode  
22C0 476 10$:  
22C0 477 :  
22C0 478 NAMESTR::  
08' 22C0 479 .BYTE 10$-5$  
0D 22C1 480 5$: .BYTE CR  
2A 23 21 22C2 481 .ASCII '!#*' ;  
0A 22C5 482 .BYTE LF ;  
43 41 21 22C6 483 .ASCII '!AC' ;  
22C9 484 10$:  
22C9 485 :  
22C9 486 :  
22C9 487 : Finish sequence -- set regular character set,  
22C9 488 : reset DEC_CRT (VT100) AVO characteristics, and  
22C9 489 : carriage return.  
22C9 490 :  
22C9 491 :  
22C9 492 FIN_SEQ::  
0C' 22C9 493 .BYTE 10$-5$  
52 1B 47 1B 22CA 494 5$: .BYTE ESC,^A/G/ ; Set regular character set  
42 1B 22CC 495 .BYTE ESC,^A/B/,ESC,^A/L/,ESC,^A/R/ ; Set AVO char's so UNDO works  
22D2 496 : ; Need to send a character to set attribs  
0D 22D2 497 .BYTE CR ; Undo DEC_CRT AVO characteristics  
55 1B 22D3 498 .BYTE ESC,^A/U/  
0D 22D5 499 .BYTE CR  
22D6 500 10$:  
22D6 501 :  
22D6 502 :
```



```

22D6 503 ; Sequence to place the cursor on the bottom line and clear it.
22D6 504 ;
22D6 505 ;
22D6 506 BOT_CURS::
01 18 59 1B 22D6 507 .BYTE 10%-5%
         4B 1B 22D7 508 5$: .BYTE ESC,^A/Y/,24,1 ; Position to bottom line on screen
         22DB 509 .BYTE ESC,^A/K/ ; Clear to end of line
22DD 510 10$:

```

```
22DD 512 :  
22DD 513 : Announcement string, including home and clear screen, set regular  
22DD 514 : char set, and identification (top line of the screen).  
22DD 515 :  
22DD 516 :  
22DD 517 ANNCE_STR::  
22DD 518 .BYTE 10%-5%  
69 6E 1D 01 59 1B 4A 1B 48 1B 47 1B 21' 22DE 519 5$: .BYTE ESC,^A/G/,ESC,^A/H/,ESC,^A/J/,ESC,^A/Y/,1,29  
79 74 6F 4D 20 53 4D 56 2F 58 41 56 22E8 520 .ASCII \VAX/VMS Monitor Utility\  
79 74 69 6C 69 74 55 20 72 6F 74 22F4  
22FF 521 10$:  
22FF 522 :  
22FF 523 :  
22FF 524 : Status string. This is bottom line (footing) of the screen.  
22FF 525 : It contains indications for 'PLAYBACK', 'SUMMARIZING',  
22FF 526 : and 'RECORDING'.  
22FF 527 :  
22FF 528 :  
22FF 529 STATUS_STR::  
01 18 59 1B 22FF 530 .BYTE 10%-5%  
43 41 21 2300 531 5$: .BYTE ESC,^A/Y/,24,1  
22 18 59 1B 2304 532 .ASCII \!AC\  
43 41 21 2307 533 .BYTE ESC,^A/Y/,24,34  
46 18 59 1B 230B 534 .ASCII \!AC\  
43 41 21 230E 535 .BYTE ESC,^A/Y/,24,70  
2312 536 .ASCII \!AC\  
2315 537 10$:  
2315 538 :  
2315 539 :  
2315 540 : Title string, including cursor positioning and title (with  
2315 541 : optional percent sign), centered and reversed.  
2315 542 :  
2315 543 :  
2315 544 TITLE_STR::  
01 02 59 1B 2315 545 .BYTE 10%-5%  
20 2A 23 21 2316 546 5$: .BYTE ESC,^A/Y/,2,1 : Position cursor  
20 29 25 28 3C 23 21 20 43 41 52 1B 231A 547 .ASCII \!#* \ : Preceding blanks  
20 29 25 28 3C 23 21 20 43 41 52 1B 231E 548 .BYTE ESC,^A/R/ : Reverse-video  
20 29 25 28 3C 23 21 20 43 41 52 1B 2320 549 .ASCII \!AC!#<(>)!>\ : Title text  
20 29 25 28 3C 23 21 20 43 41 52 1B 232C  
20 29 25 28 3C 23 21 20 43 41 52 1B 232E 550 .BYTE ESC,^A/U/ : Undo reverse-video  
41 21 20 65 64 6F 6E 20 6E 6F 20 20 2330 551 .BYTE ESC,^A/Y/,3,32 : Optional position cursor  
41 21 20 65 64 6F 6E 20 6E 6F 20 20 2334 552 .ASCII \ on node !AC\ : Optional nodename text  
43 2340  
2341 553 10$:  
2341 554 :  
2341 555 :  
2341 556 : User's comment string, including cursor positioning  
2341 557 : and comment string, centered and reversed.  
2341 558 :  
2341 559 :  
2341 560 COMM_STR::  
01 05 59 1B 2341 561 .BYTE 10%-5%  
20 2A 23 21 2342 562 5$: .BYTE ESC,^A/Y/,5,1 : Position cursor  
20 2A 23 21 2346 563 .ASCII \!#* \ : Preceding blanks  
20 46 41 21 20 234A 564 .BYTE ESC,^A/R/ : Reverse-video  
20 46 41 21 20 234C 565 .ASCII \!AF \ : Title text
```


MONITOR
V04-000

- VAX/VMS Performance Monitor Utility H.13
DECLARATIONS

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 15
(7)

55 1B 2351 566
2353 567 10\$: .BYTE ESC,^A/U/

; Undo reverse-video

```
2353 569 :  
2353 570 : Date/time string, including cursor positioning.  
2353 571 :  
2353 572 :  
2353 573 TIME_STR::  
2353 574 .BYTE 10%-5%  
1F 04 59 1B 2354 575 5$: .BYTE ESC,^A/Y/,4,31  
44 41 21 2358 576 .ASCII \!AD\  
2B 04 59 1B 235B 577 .BYTE ESC,^A/Y/,4,43  
44 41 21 235F 578 .ASCII \!AD\  
2362 579 10$:  
2362 580 :  
2362 581 :  
2362 582 : Date/time string for special SYSTEM screen  
2362 583 :  
2362 584 :  
2362 585 SYS_TIME_STR::  
2362 586 .BYTE 10%-5%  
39 01 59 1B 2363 587 5$: .BYTE ESC,^A/Y/,1,57  
44 41 21 2367 588 .ASCII \!AD\  
45 01 59 1B 236A 589 .BYTE ESC,^A/Y/,1,69  
44 41 21 236E 590 .ASCII \!AD\  
2371 591 10$:  
2371 592 :  
2371 593 :  
2371 594 : Summary line string, including cursor  
2371 595 : positioning and from/to times.  
2371 596 :  
2371 597 :  
2371 598 SUMMLINE_STR::  
2371 599 .BYTE 10%-5%  
44 41 21 20 3A 37 03 59 1B 2372 600 5$: .BYTE ESC,^A/Y/,3,55  
6D 6F 72 46 2376 601 .ASCII \From: !AD\  
25 04 59 1B 237F 602 .BYTE ESC,^A/Y/,4,37  
20 20 20 20 20 59 52 41 4D 4D 55 53 2383 603 .ASCII \SUMMARY To: !AD\  
20 20 20 3A 6F 54 20 20 20 20 20 20 238F  
44 41 21 239B  
239E 604 10$:  
239E 605 :  
239E 606 :  
239E 607 : Special summary line string for SYSTEM class  
239E 608 :  
239E 609 :  
239E 610 SYS_SUMMLINE_STR::  
239E 611 .BYTE 10%-5%  
44 41 21 20 3A 37 01 59 1B 239F 612 5$: .BYTE ESC,^A/Y/,1,55  
6D 6F 72 46 23A3 613 .ASCII \From: !AD\  
37 02 59 1B 23AC 614 .BYTE ESC,^A/Y/,2,55  
44 41 21 20 20 20 3A 6F 54 23B0 615 .ASCII \To: !AD\  
25 03 59 1B 23B9 616 .BYTE ESC,^A/Y/,3,37  
59 52 41 4D 4D 55 53 23BD 617 .ASCII \SUMMARY\  
23C4 618 10$:  
23C4 619 :  
23C4 620 PLAY_STR: ; String for footing line  
23C4 621 .BYTE 10%-5%  
20 4B 43 41 42 59 41 4C 50 20 23C5 622 5$: .BYTE ESC,^A/R/  
23C7 623 .ASCII \ PLAYBACK \ ; Reverse video
```



```
55 1B 23D1 624 .BYTE ESC,^A/U/ ; Undo the reverse
23D3 625 10$:
23D3 626
23D3 627 SUMM_STR: ; String for footing line
11' 23D3 628 .BYTE 10$-5$
52 1B 23D4 629 5$: .BYTE ESC,^A/R/ ; Reverse video
47 4E 49 5A 49 52 41 4D 4D 55 53 20 23D6 630 .ASCII \ SUMMARIZING \
20 23E2
55 1B 23E3 631 .BYTE ESC,^A/U/ ; Undo the reverse
23E3 632 10$:
23E5 633
23E5 634 REC_STR: ; String for footing line
OF' 23E5 635 .BYTE 10$-5$
52 1B 23E6 636 5$: .BYTE ESC,^A/R/ ; Reverse video
20 47 4E 49 44 52 4F 43 45 52 20 23E8 637 .ASCII \ RECORDING \
55 1B 23F3 638 .BYTE ESC,^A/U/ ; Undo the reverse
23F5 639 10$:
23F5 640
20 00' 23F5 641 BLANK_STR:: .ASCII \ \ ; Blank string for footing and heading lines
01 23F5
000023FB 23F7 642 STATUS_PARMs:: ; Status parms -- next 3 longwords
000023FF 23FB 643 FOOTP: .BLKL 1 ; Address of "playback" or blank string
00002403 23FF 644 FOOTs: .BLKL 1 ; Address of "summary" or blank string
645 FOOTR: .BLKL 1 ; Address of "record" or blank string
```

```
2403 647 :  
2403 648 : PROCESSES screen heading string.  
2403 649 :  
2403 650  
2403 651 PROCHEAD_STR::  
2403 652 .BYTE 10%-5%  
6E 75 6F 43 20 73 73 65 01 01 59 1B 2404 653 5$: .BYTE ESC,^A/Y/,1,1 ; Position cursor  
2408 654 .ASCII \Process Count:\  
2414  
2416 655 .BYTE ESC,^A/Y/,1,60 ; Position cursor  
3A 65 6D 3C 01 59 1B 241A 656 .ASCII \Uptime:\  
69 74 70 55 2421 657 .BYTE ESC,^A/Y/,6,1 ; Position cursor  
01 06 59 1B 2425 658 .BYTE ESC,^A/L/ ; Underline  
4C 1B 2427 659 .ASCII \ PID STATE PRI NAME PAGES\  
20 44 49 50 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  
20 49 52 50 20 45 54 41 54 53 20 20 20 20 20 20 20 20 20 20  
20 20 20 20 20 20 45 4D 41 4E 20 20 20 20 20 20 20 20 20 20  
53 45 47 41 50 20 20 20 20 20 20 20 20 20 20 20 20 20 20  
54 4E 43 4F 49 44 20 20 20 20 20 20 20 20 20 20 20 20 20 20  
50 43 20 20 53 54 4C 55 41 46 20 20 20 20 20 20 20 20 20 20  
20 20 20 20 20 20 45 4D 49 54 20 55 20 55 20 55 20 55 20 55  
55 1B 2477 661 .BYTE ESC,^A/U/ ; Undo underlining  
2479 662 10$:  
2479 663  
2479 664 :  
2479 665 : Tabular screen heading string.  
2479 666 :  
2479 667  
2479 668 TABHEAD_STR::  
2479 669 .BYTE 10%-5%  
52 4C 1B 247A 670 5$: .BYTE ESC,^A/L/ ; Underline  
55 43 247C 671 .ASCII \CUR\  
55 1B 247F 672 .BYTE ESC,^A/U/ ; Undo underlining  
20 20 20 20 20 20 20 20 43 41 21 2481 673 .ASCII \!AC \  
4C 1B 248B 674 .BYTE ESC,^A/L/ ; Underline  
45 56 41 248D 675 .ASCII \AVE\  
55 1B 2490 676 .BYTE ESC,^A/U/ ; Undo underlining  
20 20 20 20 20 20 20 20 43 41 21 2D 21 2492 677 .ASCII \!-!AC \  
4C 1B 249E 678 .BYTE ESC,^A/L/ ; Underline  
4E 49 4D 24A0 679 .ASCII \MIN\  
55 1B 24A3 680 .BYTE ESC,^A/U/ ; Undo underlining  
20 20 20 20 20 20 20 20 43 41 21 2D 21 24A5 681 .ASCII \!-!AC \  
4C 1B 24B1 682 .BYTE ESC,^A/L/ ; Underline  
58 41 4D 24B3 683 .ASCII \MAX\  
55 1B 24B6 684 .BYTE ESC,^A/U/ ; Undo underlining  
43 41 21 2D 21 24B8 685 .ASCII \!-!AC\  
24BD 686 10$:
```



```
24BD 688 ;
24BD 689 ; Heading string for special SYSTEM class screen
24BD 690 ;
24BD 691
24BD 692 SYS_HEAD_STR::
24BD 693 .BYTE 10$-5$
24BE 694 5$: .BYTE ESC,^A/Y/,1,1
24C2 695 .ASCII \Node: !AC\
24CB 696 .BYTE ESC,^A/Y/,2,1
24CF 697 .ASCII \Statistic: \
24DA 698 .ASCII \!AD\
24DD 699 10$:
24DD 700
24DD 701 ;
24DD 702 ; Bar graph screen heading string.
24DD 703 ;
24DD 704
24DD 705 BARHEAD_STR::
24DD 706 .BYTE 10$-5$
24DE 707 5$: .BYTE ESC,^A/Y/,6,38
24E2 708 .ASCII \!10<!UL!AC!>!10<!UL!AC!>!10<!UL!AC!>!6<!UL!AC!>!#< !>!5UL!AC\
24EE
24FA
2506
2512
251E 709 10$:
251E 710
251E 711 ;
251E 712 ; Bar graph statistic heading string (in smaller box).
251E 713 ;
251E 714
251E 715 STATHEAD_STR::
251E 716 .BYTE 10$-5$
251F 717 5$: .BYTE ESC,^A/Y/,2,13
2523 718 .ASCII \+-----+\
252A 719 .BYTE ESC,^A/Y/,3,13
252E 720 .ASCII \! \
2530 721 .ASCII \!AD\
2533 722 .ASCII \! \
2535 723 .BYTE ESC,^A/Y/,4,13
2539 724 .ASCII \+-----+\
2540 725 10$:
2540 726
2540 727 ;
2540 728 ; Other bar graph strings
2540 729 ;
2540 730
2540 731 CURSOR_STR:: .BYTE ESC,^A\Y\ ; Position cursor escape sequence
2542 732
2542 733 HORIZ_STR:: .ASCII \+ - - - - + - - - - + - - - - + - - - - +\
254E
255A
2566
256C 734 ; Top and bottom line of bar graph box
256C 735 PCENT_STR:: .ASCIC \% \ ; Percent symbol string for heading line
256C 736 K_STR:: .ASCIC \% \ ; K symbol string for heading on bar graph b
256E
```

01 01 59 1B 1F' 24BE 694 5\$: .BYTE ESC,^A/Y/,1,1
43 41 21 20 3A 65 64 6F 4E 24C2 695 .ASCII \Node: !AC\
01 02 59 1B 24CB 696 .BYTE ESC,^A/Y/,2,1
20 3A 63 69 74 73 69 74 61 74 53 24CF 697 .ASCII \Statistic: \
44 41 21 24DA 698 .ASCII \!AD\
24DD 699 10\$:
24DD 700
24DD 701 ;
24DD 702 ; Bar graph screen heading string.
24DD 703 ;
24DD 704
24DD 705 BARHEAD_STR::
24DD 706 .BYTE 10\$-5\$
24DE 707 5\$: .BYTE ESC,^A/Y/,6,38
24E2 708 .ASCII \!10<!UL!AC!>!10<!UL!AC!>!10<!UL!AC!>!6<!UL!AC!>!#< !>!5UL!AC\
24EE
24FA
2506
2512
251E 709 10\$:
251E 710
251E 711 ;
251E 712 ; Bar graph statistic heading string (in smaller box).
251E 713 ;
251E 714
251E 715 STATHEAD_STR::
251E 716 .BYTE 10\$-5\$
251F 717 5\$: .BYTE ESC,^A/Y/,2,13
2523 718 .ASCII \+-----+\
252A 719 .BYTE ESC,^A/Y/,3,13
252E 720 .ASCII \! \
2530 721 .ASCII \!AD\
2533 722 .ASCII \! \
2535 723 .BYTE ESC,^A/Y/,4,13
2539 724 .ASCII \+-----+\
2540 725 10\$:
2540 726
2540 727 ;
2540 728 ; Other bar graph strings
2540 729 ;
2540 730
2540 731 CURSOR_STR:: .BYTE ESC,^A\Y\ ; Position cursor escape sequence
2542 732
2542 733 HORIZ_STR:: .ASCII \+ - - - - + - - - - + - - - - + - - - - +\
254E
255A
2566
256C 734 ; Top and bottom line of bar graph box
256C 735 PCENT_STR:: .ASCIC \% \ ; Percent symbol string for heading line
256C 736 K_STR:: .ASCIC \% \ ; K symbol string for heading on bar graph b
256E

```

01 256E
52 55 43 2570 737 STAT_HEAD:: .ASCII \CUR\ ; Table of statistic headings for bar graph
45 56 41 2573 738 .ASCII \AVE\
4E 49 4D 2576 739 .ASCII \MIN\
58 41 4D 2579 740 .ASCII \MAX\
54 4E 45 52 52 55 43 257C 741 STAT_LONG:: .ASCII \CURRENT\ ; Long version for SYSTEM class screen
45 47 41 52 45 56 41 2583 742 .ASCII \AVERAGE\
4D 55 4D 49 4E 49 4D 258A 743 .ASCII \MINIMUM\
4D 55 4D 49 58 41 4D 2591 744 .ASCII \MAXIMUM\

```



```
2598 746 :  
2598 747 : FAO control strings for name string displays for homogeneous classes.  
2598 748 :  
2598 749 :  
43 41 21 3C 23 21 000025A0'010E0000' 2598 750 DISK_FAO: .ASCID \!#<!AC!#*#!AC!ZW:!!>!#* !AF\  
21 3A 57 5A 21 43 41 21 24 2A 23 21 25A6  
46 41 21 20 2A 23 21 3E 25B2  
21 24 3C 33 31 21 000025C2'010E0000' 25BA 751 : Disk FAO string (no alloc class)  
21 20 3A 57 5A 21 43 41 21 24 42 5A 25BA 752 DISK_FAO_AC: .ASCID \!13<$!ZB$!AC!ZW: !!>!#* !#<(!AC)!>!AF\  
43 41 21 28 3C 23 21 20 2A 23 21 3E 25C8  
46 41 21 3E 21 29 25D4  
25E0  
43 41 21 000025EE'010E0000' 25E6 753 : Disk class FAO string (with alloc cls)  
64 6F 4E 20 6E 77 6F 6E 6B 6E 55 00' 25E6 754 SCS_FAO:: .ASCID \!AC\ : SCS class FAO control string  
65 25F1 755 UNKNOWN_NODE: .ASCIC /Unknown Node/ : Counted ASCII string for a 0 length  
0C 25FD  
25FE 756 : node name in the system block.  
25FE 757 :  
25FE 758 : FAO control string for item name display for homogeneous classes  
25FE 759 :  
25FE 760  
25FE 761 ITEM_NAM_STR:  
25FE 762 .BYTE 10$-5$  
25FF 763 5$: .BYTE ESC,^A/Y/,0,1 : Cursor position  
36 32 21 3E 21 20 20 20 20 3C 23 21 2603 764 .ASCII \!#< !!>!26<!AC!>\  
3E 21 43 41 21 3C 260F  
2615 765 10$:  
00000006 2615 766 ILN_REG = 6 : Item line number for regular displays  
00002601 2615 767 ITM[ENNO = ITEM_NAM_STR+3 : Label for item line number
```

```
00002616 2615 769 SYSOUT_TYPE: .BLKB 1 ; SYSS$OUTPUT device type
2616 770 ; MONITOR SYSOUT types are:
2616 771 ; DEC CRT, VT5X, OTHER VID, HARDCOPY
2616 772 ; (Type codes defined above)
2616 773
00000013' 2616 774 PROC_SETUP_STR: .LONG 20$-10$ ; Descriptor for PROCESSES screen setup ...
0000261E' 261A 775 .LONG 10$ ; ... string (incl. cursor positioning)
261E 776 10$:
10 01 59 1B 261E 777 .BYTE ESC,^A/Y/,1,16 ; Cursor position to process count field
3E 21 4C 55 21 3C 35 21 2622 778 .ASCII \!5<!UL!>\
44 01 59 1B 262A 779 .BYTE ESC,^A/Y/,1,68 ; Cursor position to uptime field
53 41 21 262E 780 .ASCII \!A$
2631 781 20$:
2631 782
0000003D' 2631 783 PROC_RES_STR: .LONG 20$-10$ ; Descriptor for PROCESSES FAO control ...
00002639' 2635 784 .LONG 10$ ; ... string (resident process)
41 35 21 20 4C 58 21 20 20 20 20 20 20 20 20 20 2639 785 10$: .ASCII \ !XL !5AC !2UL !4(+)!15AF !9<!UL!/!UL!> !7UL !7UL !2(+)!
29 2B 28 34 21 20 4C 55 32 21 20 43 2645
4C 55 21 3C 39 21 20 46 41 35 31 21 2651
20 4C 55 37 21 20 3E 21 4C 55 21 2F 265D
25 21 29 2B 28 32 21 20 4C 55 37 21 2669
54 2675
2676 786 20$:
2676 787
00000045' 2676 788 PROC_NRES_STR: .LONG 20$-10$ ; Descriptor for PROCESSES FAO control ...
0000267E' 267A 789 .LONG 10$ ; ... string (non-resident process)
41 35 21 20 4C 58 21 20 20 20 20 20 20 20 20 20 267E 790 10$: .ASCII \ !XL !5AC !2UL !4(+)!15AF !9<!UL!/!UL!>!5(+) SWAPP
29 2B 28 34 21 20 4C 55 32 21 20 43 268A
4C 55 21 3C 39 21 20 46 41 35 31 21 2696
20 29 2B 28 35 21 3E 21 4C 55 21 2F 26A2
45 50 50 41 57 53 20 20 20 20 20 20 20 20 20 20 26AE
20 2A 39 21 54 55 4F 20 44 26BA
26C3 791 20$:
26C3 792
26C3 793
26C3 794 ; Top PROCESSES FAO control string for one process display line
26C3 795
26C3 796
26C3 797 TOPSTR:
26C3 798 .BYTE 10$-5$
26C4 799 5$: .BYTE ESC,^A/Y/,0,2 ; position to left margin
31 21 29 2B 28 34 21 20 20 4C 58 21 26C8 800 .ASCII \!XL !4(+)!15AF !7<!#UL!> \
55 23 21 3C 37 21 20 20 20 46 41 35 26D4
20 20 3E 21 4C 26E0
46 1B 26E5
2A 2A 23 21 26E7
4B 1B 47 1B 26EB
26EF 801 .BYTE ESC,^A/F/ ; select alternate char set
26EF 802 .ASCII \!#**\ ; repeating bar character
26EF 803 .BYTE ESC,^A/G/,ESC,^A/K/ ; select reg set and erase to EOL
26EF 804 10$:
26EF 805 TOPLNNO = TOPSTR+3 ; label for line number
000026C6 26EF 806 TOPBAR = TOPSTR+39 ; label for bar character
000026EA 26EF
26EF 807
26EF 808
26EF 809 ; Top PROCESSES FAO control string to erase a line
26EF 810
26EF 811
26EF 812 ERLINE_STR:
06' 26EF 813 .BYTE 10$-5$
```



```
01 00 59 1B 26F0 814 5$: .BYTE ESC,^A/Y/,0,1 ; position to left margin
      4B 1B 26F4 815 .BYTE ESC,^A/K/ ; erase to end of line
      26F6 816 10$: ;
000026F2 26F6 817 ERLNNO = ERLINE_STR+3 ; label for line number
      26F6 818 ;
00000004' 26F6 819 VT100_REGSET: .LONG 20$-10$ ; descriptor for VT100 ...
000026FE' 26FA 820 .LONG 10$ ; ... "regular" char set esc seq
OF 42 28 1B 26FE 821 10$: .BYTE ESC,^A/(/,^A/B/,SI
      2702 822 20$: ;
00000003' 2702 823 VT100_ALTSET: .LONG 20$-10$ ; descriptor for VT100 ...
0000270A' 2706 824 .LONG 10$ ; ... "alternate" graphics set esc seq
30 28 1B 270A 825 10$: .BYTE ESC,^A/(/,^A/O/
      270D 826 20$: ;
00002711' 270D 827 VT100_CURSET: .BLKL 1 ; addr of esc seq descr for curr char set
00000002' 2711 828 REG_SET: .LONG 20$-10$ ; VT52 esc seq to estab regular char set
00002719' 2715 829 .LONG 10$
      47 1B 2719 830 10$: .BYTE ESC,^A/G/
      271B 831 20$: ;
      271B 832 ;
00000004' 271B 833 PUTSCRARG: ; arg list for PUT_SCREEN call
0000272F' 271F 834 .LONG 4 ; argument count
00000000 2723 835 .LONG TXT_DESC ; addr of buffer to display
00000000 272B 836 .LONG 0,0 ; no cursor pos specification
      272F 837 ATTRIBMSK: .LONG 0 ; start off with no special attributes
      272F 838 ;
00002733' 272F 839 TXT_DESC: ; hold area for descriptor to be PUT'd
00002737' 2733 840 TXT_LENGTH: .BLKL 1 ; length
      2737 841 TXT_START: .BLKL 1 ; address
59 55 52 4B 4A 48 47 46 42 4C 2737 842 ;
0000000A 2741 843 ESC_SEQ_TABLE: .ASCII \LBFGHJKRUY\ ; table of valid escape modifiers
      2741 844 ES_TAB_LEN = .-ESC_SEQ_TABLE ; length of table
000027A9 2741 845 ;
      2741 846 PUTMSGVEC: .BLKL PUTMSGSIZE ; Message argument vector for $PUTMSG
```

```
27A9 848 .SBTTL CALC_LEN - Calculate class record lengths
0000 849 .PSECT $$MONCODE,NOWRT,EXE
      850 :++
      851 :
      852 : FUNCTIONAL DESCRIPTION:
      853 :
      854 : This routine is called to calculate the length of a block
      855 : (CDB$W_BLKLEN) for each STANDARD class. Non-standard
      856 : classes have this value entered at compile-time. Block
      857 : length for standard heterogeneous classes is defined as the
      858 : sum of the sizes of all data items which comprise the class,
      859 : and which are recorded (calculated items which are displayed but
      860 : not recorded are not included in block length).
      861 : Block length for standard homogeneous classes is the sum of
      862 : all data items comprising the class (including the element ID)
      863 : for a SINGLE element (e.g., for a single disk).
      864 :
      865 : In addition, some pre-processing for the MODES class is done,
      866 : and, for homogeneous classes, the CDX$W_CUMELCT, CDX$B_IDISCONSEC
      867 : CDX$W_IBITS and CDX$B_IDISCT fields are initialized.
      868 :
      869 : CALLING SEQUENCE:
      870 :
      871 : CALLS #1,CALC_LEN
      872 :
      873 : INPUTS:
      874 :
      875 : 4(AP) - address of MRB$O_CLASSBITS, the bit string
      876 :         representing classes to be monitored.
      877 :
      878 : IMPLICIT INPUTS:
      879 :
      880 : PERFTABLE - table of IDB's describing each data item,
      881 :             indexed by item number ( * entry size).
      882 :
      883 : CDBHEAD - table of CDB's, one for each class.
      884 :
      885 : MAX_CLASS_NO - maximum class number (class numbers are zero-origin)
      886 :
      887 : MODES_CLSNO - MODES class number
      888 :
      889 : MODES_ICOUNT - MODES item count (for uniprocessor)
      890 :
      891 : OUTPUTS:
      892 :
      893 : None
      894 :
      895 : IMPLICIT OUTPUTS:
      896 :
      897 : CDB$W_BLKLEN (block length) field established for each CDB.
      898 :
      899 : PROCS_PER_REC field established for the PROCESSES class.
      900 :
      901 :
      902 : ROUTINE VALUE:
      903 :
      904 : RO = $$$_NORMAL, or MNR$_ITMNOTDEF
```



```
0000 905 ;
0000 906 ; SIDE EFFECTS:
0000 907 ;
0000 908 ; None
0000 909 ;
0000 910 ; --
0000 911 ;
0000 912 ;
OF5C 0000 913 .ENTRY CALC_LEN, ^M<R2,R3,R4,R6,R8,R9,R10,R11>
0002 914 ;
0002 915 ;
0002 916 ; First, re-establish item count for the MODES class (uniprocessor)
0002 917 ;
0002 918 ;
16 04 BC 00000000'8F E1 0002 919 BBC #MODES_CLSNO,@4(AP),10$ ; Skip if MODES not present
56 00000000'EF DE 000B 920 MOVAL CDBHEAD,R6 ; Get address of first CDB
56 00000000'8F C0 0012 921 ADDL #<CDB$K_SIZE*MODES_CLSNO>,R6 ; Calculate addr of MODES CDB
14 A6 00000000'8F D0 0019 922 MOVL #MODES_ICOUNT,CDB$_ICOUNT(R6) ; Get uniprocessor item count
0021 923 ;
0021 924 ;
0021 925 ; Now calculate CDB$_BLKLEN for all requested classes.
0021 926 ;
0021 927 ;
0021 928 10$:
5B D4 0021 929 CLRL R11 ; Init starting bit position
59 20 D0 0023 930 20$:
58 5B D0 0026 931 MOVL #32,R9 ; Init bit field size
0026 932 ; NOTE -- must handle in 32-bit chunks
0029 933 MOVL R11,R8 ; Init start position of next chunk
5A 04 BC 59 58 EA 0029 934 30$:
11 13 002F 935 FFS R8,R9,@4(AP),R10 ; Search for next class number
18 10 0031 936 ; R10 contains class no. if found
14 50 E9 0033 937 BEQL 40$ ; Branch if none found this chunk
59 58 C0 0036 938 BSBB CALC_CLASS ; Calc block length for this class
5A 01 C1 0039 939 BLBC R0,50$ ; Go return if error
59 58 C2 003D 940 ADDL2 R8,R9 ; Compute next starting
E7 11 0040 941 ADDL3 #1,R10,R8 ; ... position and field size
0040 942 SUBL2 R8,R9 ; ... for this chunk
0042 943 BRB 30$ ; Go search rest of chunk
FFD9 5B 20 0000'8F 3D 0042 944 40$:
004A 945 ACBW #MAX_CLASS_NO,#32,R11,20$ ; Loop to process next chunk
004A 946 ;
004A 947 ;
004A 948 ; At this point, CDB$_BLKLEN fields for all monitored classes
004A 949 ; have been established.
004A 950 ;
004A 951 ;
004A 952 50$:
04 004A 953 RET ; Return with status in R0
004B 954 ;
004B 955 ;
004B 956 CALC_CLASS: ; Calc block length for this class
004B 957 ; NOTE -- R10 contains class number
004B 958 ; Regs R8 thru R11 must not be changed
56 5A 00000053 8F C5 004B 959 ;
56 00000000'EF46 9E 0053 960 MULL3 #CDB$K_SIZE,R10,R6 ; Compute offset to desired CDB
961 MOVAB CDBHEAD[R6],R6 ; Index to CDB address
```

```
005B 962
005B 963
005B 964 : For all classes except homogeneous standard classes, set the
005B 965 : element count (of elements displayed) equal to the item count
005B 966 : (of items collected).
005B 967
005B 968
05 4B A6 05 E0 005B 969 BBS #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if homog
18 A6 14 A6 D0 0060 970 MOVL CDB$L_ICOUNT(R6),CDB$L_ECOUNTR6) ; Item count = elt count
0065 971 10$:
10 4B A6 04 E0 0065 972 BBS #CDB$V_STD,CDB$L_FLAGS(R6),20$ ; Br if a standard class
52 00007CEB 8F D0 006A 973 MOVL #<MAX_REC_SIZE-MNR_CLSSK_HSIZE-MNR_PROSK_PSIZE>,R2
0016'CF 52 20 A6 A7 0071 974 : Get max data size
0071 975 DIVW3 CDB$W_BLKLEN(R6),R2,W^PROCS_PER_REC
0078 976 : Compute processes per record for ...
0078 977 : ... PROCESSES non-STD class
0078 978 : All done with this class
007A 979 20$:
50 14 A6 D0 007A 980 MOVL CDB$L_ICOUNT(R6),R0 ; Get no of items to sum for this CDB
51 1C A6 D0 007E 981 MOVL CDB$A_ITMSTR(R6),R1 ; Address of item-number string
52 D4 0082 982 CLRL R2 ; Clear block size reg
0084 983 30$:
54 81 9A 0084 984 MOVZBL (R1)+,R4 ; Get next item number
54 11 C4 0087 985 MULL #IDB$K_ILENGTH,R4 ; Compute index into IDB table
54 0000'CF44 9E 008A 986 MOVAB W^PERFTABLE[R4],R4 ; Address of IDB for this item
10 A4 95 0090 987 TSTB IDB$B_FLAGS(R4) ; Is this a calculated item?
17 12 0093 988 BNEQ 70$ ; Branch if so (don't add to size)
0095 989 CASE IDB$W_ISIZE(R4),<40$,50$,60$>,W ; Select on proper size
00A0 990
52 D6 00A0 991 40$: INCL R2 ; Add 1 for byte
08 11 00A2 992 BRB 70$
52 02 C0 00A4 993 50$: ADDL #2,R2 ; Add 2 for word
03 11 00A7 994 BRB 70$
52 04 C0 00A9 995 60$: ADDL #4,R2 ; Add 4 for longword
D5 50 F5 00AC 996 70$: SOBGTR R0,30$ ; Loop for each item in this class
00AF 997
20 A6 52 B0 00AF 998 MOVW R2,CDB$W_BLKLEN(R6) ; Store away size for this class
00B3 999
00B3 1000 :
00B3 1001 : Now add in size of element ID for homogeneous classes
00B3 1002 :
00B3 1003 :
16 4B A6 05 E1 00B3 1004 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),80$ ; All done if hetero
52 32 A6 D0 00B8 1005 MOVL CDB$A_CDX(R6),R2 ; Get CDX addr for homog class
50 09 A2 9A 00BC 1006 MOVZBL CDX$B_ELIDLEN(R2),R0 ; Get length of element ID
20 A6 50 A0 00C0 1007 ADDW2 R0,CDB$W_BLKLEN(R6) ; Add it in to get data block size
00C4 1008 :
00C4 1009 : Also, for homogeneous classes, initialize CDX$W_CUMELCT
00C4 1010 : and CDX$B_IDISCONSEC, and calculate display item count.
00C4 1011 :
00C4 1012 :
0A A2 B4 00C4 1013 CLRW CDX$W_CUMELCT(R2) ; Init cumulative element count
07 A2 94 00C7 1014 CLRB CDX$B_IDISCONSEC(R2) ; Init consecutive display number
00CA 1015
06 10 00CA 1016 BSBB CALC_DITEM ; Calculate display item count
03 11 00CC 1017 BRB 90$ ; Go return with status in R0
00CE 1018 80$:
```


MONITOR
V04-000

- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
CALC_LEN - Calculate class record length 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 27
(13)

```
50 01 D0 00CE 1019      MOVL  #SS$_NORMAL,R0      ; Success status
      00D1 1020
      05 00D1 1021 90$:  RSB      ; Return with status in R0
```

```
00D2 1023 CALC_DITEM: ; Calc display item cnt for homog class
00D2 1024
50 14 A6 D0 00D2 1025 MOVL CDB$$_ICOUNT(R6),R0 ; Get count of all items for class
26 62 OF E0 00D6 1026 BBS #CDX$$_IBITS-1, - ; Br if ALL items requested for display
00DA 1027 CDX$_IBITS(R2),30$
00DA 1028
00DA 1029 ;
00DA 1030 ; Use FFS loop to calculate the number of display items requested.
00DA 1031 ; Store the number in CDX$_IDISCT. Also, clear any bits higher
00DA 1032 ; than the number of items defined for this revision level.
00DA 1033 ;
00DA 1034
62 51 10 50 83 00DA 1035 SUBB3 R0,#CDX$$_IBITS,R1 ; Calc number of unused bits
51 50 00 F0 00DE 1036 INSV #0,R0,R1,CDX$_IBITS(R2) ; ... and make sure they are clear
54 D4 00E3 1037 CLRL R4 ; Init counter of requested items
51 D4 00E5 1038 CLRL R1 ; Init starting bit number for FFS
00E7 1039 10$: FFS R1,R0,CDX$_IBITS(R2),R3 ; Search for next item number
53 62 50 51 EA 00E7 1040 ; R3 contains item number if found
00EC 1041 ; Branch if none found
00EC 1042
0C 13 00EC 1043 BEQL 20$
54 D6 00EE 1044 INCL R4 ; Count this item
00F0 1045
51 53 01 C1 00F0 1046 ADDL3 #1,R3,R1 ; Compute next starting ...
50 10 51 C3 00F4 1047 SUBL3 R1,#CDX$$_IBITS,R0 ; ... position and field size
ED 11 00F8 1048 BRB 10$ ; Go search rest of bit string
00FA 1049
00FA 1050 20$:
06 A2 54 90 00FA 1051 MOVVB R4,CDX$_IDISCT(R2) ; Store number of display items
OF 11 00FE 1052 BRB 40$ ; ... and go return
0100 1053
0100 1054 ;
0100 1055 ; ALL items requested for display. Store number requested in
0100 1056 ; CDX$_IDISCT, and set all item bits in CDX$_IBITS.
0100 1057 ;
0100 1058 ;
0100 1059
0100 1060 30$:
62 50 00 FFFFFFFF 62 B4 0100 1061 CLRW CDX$_IBITS(R2) ; Start out with all item bits clear
06 A2 50 F0 0102 1062 INSV #-1,#0,R0,CDX$_IBITS(R2) ; Set all bits defined for this rev.
010B 1063 MOVVB R0,CDX$_IDISCT(R2) ; ... and store its count
010F 1064 40$:
50 01 D0 010F 1065
62 B5 0112 1066 MOVL #$$$_NORMAL,R0 ; Assume normal status
14 12 0114 1067 TSTW CDX$_IBITS(R2) ; Check if no items requested
0116 1068 BNEQU 50$ ; Br if at least one requested
0116 1069
00000000'8F DD 0116 1070 PUSHL #MNR$_ITMNOTDEF ; Stack MONITOR failing status code
00001EB6'EF 01 FB 011C 1071 CALLS #1,MON_ERR ; Log the error
50 00000000'8F D0 0123 1072 MOVL #MNR$_ITMNOTDEF,R0 ; Get status to caller
012A 1073 50$:
05 012A 1074 RSB ; Return to caller
```



```
012B 1076 .SBTTL MOVE_CLASS_QUALS - Move Class Qualifier Values
012B 1077 :++
012B 1078 :
012B 1079 : FUNCTIONAL DESCRIPTION:
012B 1080 :
012B 1081 : This routine is called to move a set of values from
012B 1082 : one CDB or CDX field to another. In particular, the QFLAGS
012B 1083 : (class qualifier flags), IBITS (item bits) and the ST
012B 1084 : (display statistic) values are moved among three fields
012B 1085 : defined for each, representing default value, current value
012B 1086 : and active value. The types of moves are defined below under
012B 1087 : INPUTS.
012B 1088 :
012B 1089 : CALLING SEQUENCE:
012B 1090 :
012B 1091 : CALLS #1,MOVE_CLASS_QUALS
012B 1092 :
012B 1093 : INPUTS:
012B 1094 :
012B 1095 : 4(AP) - address of a byte containing a code indicating which
012B 1096 : type of move to make, as follows:
012B 1097 :
012B 1098 : If code =
012B 1099 :
012B 1100 : DEF_TO_CUR(=0) => Move default values to current values.
012B 1101 :
012B 1102 : CUR_TO_ACT(=1) => Move current values to active values.
012B 1103 : In addition, clear the CDB$V_EXPLICIT bit.
012B 1104 :
012B 1105 : ACT_TO_CUR(=2) => Move active values to current values.
012B 1106 :
012B 1107 : ALL_TO_ACT(=3) => Move the ALL statistic value to active.
012B 1108 :
012B 1109 : IMPLICIT INPUTS:
012B 1110 :
012B 1111 : CDBHEAD - table of CDB's, one for each class.
012B 1112 :
012B 1113 : MAX_CLASS_NO - maximum class number (class numbers are zero-origin)
012B 1114 :
012B 1115 : CDB$B_ST, CDB$B_ST_DEF and CDB$B_ST_CUR fields for each CDB.
012B 1116 :
012B 1117 : CDB$W_QFLAGS, CDB$W_QFLAGS_DEF and CDB$W_QFLAGS_CUR fields for each CDB.
012B 1118 :
012B 1119 : CDX$W_IBITS, CDX$W_IBITS_DEF and CDX$W_IBITS_CUR fields for each homog class
012B 1120 :
012B 1121 : OUTPUTS:
012B 1122 :
012B 1123 : None
012B 1124 :
012B 1125 : IMPLICIT OUTPUTS:
012B 1126 :
012B 1127 : Requested move is performed.
012B 1128 :
012B 1129 : ROUTINE VALUE:
012B 1130 :
012B 1131 : R0 = SSS_NORMAL
012B 1132 :
```

```
012B 1133 : SIDE EFFECTS:
012B 1134 :
012B 1135 : None
012B 1136 :
012B 1137 : --
012B 1138 :
012B 1139 :
0048 012B 1140 .ENTRY MOVE_CLASS_QUALS, ^M<R3,R6>
012D 1141
56 00000000'EF DE 012D 1142 MOVAL CDBHEAD,R6 ; Get address of first CDB
53 00000001'8F D0 0134 1143 MOVL #MAX CLASS NO+1,R3 ; Get number of CDB's
0090 31 013B 1144 CASE @4(AP),<10$,20$,30$,40$>,B ; Select on type of move
0148 1145 BRW 50$ ; Do nothing if out of range
014B 1146
014B 1147 :
014B 1148 : DEFAULT TO CURRENT
014B 1149 :
014B 1150 :
014B 1151 10$:
44 A6 43 A6 90 014B 1152 MOV B CDB$B-ST_DEF(R6),CDB$B-ST_CUR(R6) ; Load default stat to cur
49 A6 47 A6 B0 0150 1153 MOV B CDB$W-QFLAGS_DEF(R6),CDB$W-QFLAGS_CUR(R6) ; Load default qual flags
09 4B A6 05 E1 0155 1154 BBC #CDB$V_HOMOG,CDB$W-QFLAGS_CUR(R6) ; Br if heterogeneous clas
50 32 A6 D0 015A 1155 MOVL CDB$A_CDX(R6),R0 ; Get CDX address
04 A0 02 A0 B0 015E 1156 MOV B CDB$W-IBITS_DEF(R0),CDB$W-IBITS_CUR(R0) ; Load def item bits to cu
56 00000053 8F C0 0163 1157 15$: ADDL #CDB$K_SIZE,R6 ; Point to CDB for next cl
DE 53 F5 016A 1158 SOBGTR R3,10$ ; Loop for each CDB
006B 31 016D 1159 BRW 50$ ; Go to common exit
0170 1160
0170 1161 :
0170 1162 : CURRENT TO ACTIVE
0170 1163 :
0170 1164 :
0170 1165 :
0170 1166 20$:
42 A6 44 A6 90 0170 1167 MOV B CDB$B-ST_CUR(R6),CDB$B-ST(R6) ; Load current stat to act
45 A6 49 A6 B0 0175 1168 MOV B CDB$W-QFLAGS_CUR(R6),CDB$W-QFLAGS(R6) ; Load current qual flags
00 4B A6 0C E5 017A 1169 BBCC #CDB$V_EXPLICIT,CDB$W-QFLAGS(R6),22$ ; Indicate no explicit qua
017F 1170 22$:
08 4B A6 05 E1 017F 1171 BBC #CDB$V_HOMOG,CDB$W-QFLAGS_CUR(R6),25$ ; Br if heterogeneous clas
50 32 A6 D0 0184 1172 MOVL CDB$A_CDX(R6),R0 ; Get CDX address
60 04 A0 B0 0188 1173 MOV B CDB$W-IBITS_CUR(R0),CDB$W-IBITS(R0) ; Load curr item bits to a
018C 1174 25$:
56 00000053 8F C0 018C 1175 ADDL #CDB$K_SIZE,R6 ; Point to CDB for next class
DA 53 F5 0193 1176 SOBGTR R3,20$ ; Loop for each CDB
0042 31 0196 1177 BRW 50$ ; Go to common exit
0199 1178
0199 1179 :
0199 1180 : ACTIVE TO CURRENT
0199 1181 :
0199 1182 :
0199 1183 :
0199 1184 30$:
44 A6 42 A6 90 0199 1185 MOV B CDB$B-ST(R6),CDB$B-ST_CUR(R6) ; Load active stat back to
49 A6 45 A6 B0 019E 1186 MOV B CDB$W-QFLAGS(R6),CDB$W-QFLAGS_CUR(R6) ; Load active qual flags b
08 4B A6 05 E1 01A3 1187 BBC #CDB$V_HOMOG,CDB$W-QFLAGS_CUR(R6),35$ ; Br if heterogeneous clas
50 32 A6 D0 01A8 1188 MOVL CDB$A_CDX(R6),R0 ; Get CDX address
04 A0 60 B0 01AC 1189 MOV B CDB$W-IBITS(R0),CDB$W-IBITS_CUR(R0) ; Load active item bits ba
```



```

01B0 1190 35$:
01B0 1191
56 00000053 8F C0 01B0 1192 ADDL #CDB$K_SIZE,R6 ; Point to CDB for next class
    DF 53 F5 01B7 1193 SOBGTR R3,30$ ; Loop for each CDB
    1F 11 01BA 1194 BRB 50$ ; Go to common exit
    01BC 1195
    01BC 1196 ;
    01BC 1197 ; /ALL TO ACTIVE
    01BC 1198 ;
    01BC 1199
    01BC 1200 40$:
10 4B A6 04 E1 01BC 1201 BBC #CDB$V STD,CDB$L_FLAGS(R6),45$ ; Br if non-standard class
44 A6 43 A6 91 01C1 1202 CMPB CDB$B_ST_DEF(R6),CDB$B_ST_CUR(R6) ; Is default stat equal to
    09 12 01C6 1203 BNEQ 45$ ; Br if not
04 4B A6 0C E0 01C8 1204 BBS #CDB$V_EXPLIC,CDB$L_FLAGS(R6),45$ ; Br if explicit qualifier
42 A6 00 90 01CD 1205 MOVB #ALL_STAT,CDB$B_ST(R6) ; Force the ALL statistic
56 00000053 8F C0 01D1 1206 45$:
    E1 53 F5 01D1 1207 ADDL #CDB$K_SIZE,R6 ; Point to CDB for next class
    01DB 1208 SOBGTR R3,40$ ; Loop for each CDB
    01DB 1209
    01DB 1210 50$:
    50 01 D0 01DB 1211 MOVL #SS$_NORMAL,R0 ; Indicate success
    04 01DE 1212 RET ; ... and return
```

```
01DF 1214 .SBTTL FETCH - Collect Data into Buffer
01DF 1215
01DF 1216 :++
01DF 1217
01DF 1218 : FUNCTIONAL DESCRIPTION:
01DF 1219
01DF 1220 : This routine is called to collect the data for the next interval.
01DF 1221 : It scans a table describing which items to collect, and moves
01DF 1222 : each item to the proper slot in the collection buffer supplied
01DF 1223 : by the caller.
01DF 1224
01DF 1225 : CALLING SEQUENCE:
01DF 1226
01DF 1227 : Entered via CALL from $CMKRNL system service.
01DF 1228
01DF 1229 : INPUTS:
01DF 1230
01DF 1231 : 4(AP) - address of CDB (Class Descriptor Block)
01DF 1232
01DF 1233 : 8(AP) - address of 1st byte of variable portion of collection buffer
01DF 1234
01DF 1235 : IMPLICIT INPUTS:
01DF 1236
01DF 1237 : EXE$GQ_SYSTIME - current time in system time (quadword) units
01DF 1238
01DF 1239 : PERFTABLE - table describing each data item, indexed by
01DF 1240 : item number ( * entry size)
01DF 1241
01DF 1242 : OUTPUTS:
01DF 1243
01DF 1244 : None
01DF 1245
01DF 1246 : IMPLICIT OUTPUTS:
01DF 1247
01DF 1248 : CURRENT collection buffer is filled with raw data.
01DF 1249
01DF 1250 : ROUTINE VALUE:
01DF 1251
01DF 1252 : $$$_NORMAL
01DF 1253
01DF 1254 : SIDE EFFECTS:
01DF 1255
01DF 1256 : None
01DF 1257
01DF 1258 :--
01DF 1259
0078 01DF 1260
01E1 1261 .ENTRY FETCH, ^M<R3,R4,R5,R6>
01E1 1262
01E5 1263 : MOVL 4(AP),R6 ; Load CDB pointer
01E9 1264 : MOVL 8(AP),R5 ; Load addr of 1st byte of actual data
01F1 1265 : MOVQ G^EXE$GQ_SYSTIME,<MNR_CLSSQ_STAMP-MNR_CLSSK_HSIZE>(R5) ; Get current time into coll buffer
01F1 1266
01F1 1267 :
01F1 1268 : If this class has a pre-collection routine, call it.
01F1 1269 :
01F1 1270 : TSTL CDB$A_PRECOLL(R6) ; Is there a pre-collection rtn?
```

56 04 AC D0
55 08 AC D0
F6 A5 00000000'GF 7D
22 A6 D5


```

      0C 13 01F4 1271      BEQL 10$      ; No -- continue
      55 DD 01F6 1272      PUSHL R5      ; Yes -- stack coll buffer addr
22 B6 01 FB 01F8 1273      CALLS #1,@CDB$A_PRECOLL(R6) ; Call it
      58 50 E9 01FC 1274      BLBC R0,70$ ; If failed, get out
      46 51 E9 01FF 1275      BLBC R1,65$ ; If fetch is not required, skip it
      0202 1276 10$:      BPC #CDB$V_STD,CDB$L_FLAGS(R6),65$ ; Skip fetch if non-STD class
3C 4B A6 04 E1 0202 1277      BBS #CDB$V_HOMOG,CDB$L_FLAGS(R6),65$ ; Skip fetch if homog class
      05 E0 0207 1278
      020C 1279
      50 14 A6 D0 020C 1280      MOVL CDB$L_ICOUNT(R6),R0 ; Get number of items to fetch
      51 1C A6 D0 0210 1281      MOVL CDB$A_ITMSTR(R6),R1 ; Address of item-number string
      53 D4 0214 1282      CLRL R3 ; Clear loop counter
      0216 1283 20$:
      54 81 9A 0216 1284      MOVZBL (R1)+,R4 ; Get next item number
      54 11 C4 0219 1285      MULL #IDB$K_ILENGTH,R4 ; Compute index into IDB table
54 0000 CF 44 9E 021C 1286      MOVAB W^PERFTABLE[R4],R4 ; Address of IDB for this item
      10 A4 95 0222 1287      TSTB IDB$B_FLAGS(R4) ; Is this a computed item?
      02 13 0225 1288      BEQL 25$ ; no, go collect
      1B 11 0227 1289      BRB 60$ ; yes, skip this item and on to the next
      0229 1290 25$:
      0229 1291      CASE IDB$W_ISIZE(R4),<30$,40$,50$>,W ; Select on proper size
      0234 1292
      85 0C B4 90 0234 1293 30$:      MOVB @IDB$A_ADDR(R4),(R5)+ ; Collect a byte
      0A 11 0238 1294      BRB 60$
      85 0C B4 B0 023A 1295 40$:      MOVW @IDB$A_ADDR(R4),(R5)+ ; Collect a word
      04 11 023E 1296      BRB 60$
      85 0C B4 D0 0240 1297 50$:      MOVL @IDB$A_ADDR(R4),(R5)+ ; Collect a longword
      CE 53 50 F2 0244 1298 60$:      AOBLSS R0,R3,20$ ; Loop until done
      0248 1299
      0248 1300 ;
      0248 1301 ; If this class has a post-collection routine, call it.
      0248 1302 ;
      0248 1303
      0248 1304 65$:
      50 01 D0 0248 1305      MOVL #SS$ NORMAL,R0 ; Assume good status at this point
      26 A6 D5 024B 1306      TSTL CDB$A_POSTCOLL(R6) ; Is there a post-collection rtn?
      07 13 024E 1307      BEQL 70$ ; No -- go return
      08 AC DD 0250 1308      PUSHL 8(AP) ; Yes -- stack coll buffer addr
26 B6 01 FB 0253 1309      CALLS #1,@CDB$A_POSTCOLL(R6) ; Call it
      0257 1310
      0257 1311 70$:
      04 0257 1312      RET ; Return with status
```

```
0258 1314 .SBTTL CLASS_COLLECT - Collect & Transform Data
0258 1315
0258 1316 :++
0258 1317 :
0258 1318 : FUNCTIONAL DESCRIPTION:
0258 1319 :
0258 1320 : This routine is called once per interval - class to collect
0258 1321 : and record raw data and to do statistical sformations of
0258 1322 : that data. The transformations include ca ions of minimum
0258 1323 : value, maximum value, total since request an, percent, etc.
0258 1324 : On the first call to this routine per request, up to ten buffers
0258 1325 : are obtained. These consist of two flip-flopped collection
0258 1326 : buffers for raw data, and up to 8 statistics buffers. On subsequent
0258 1327 : calls, the buffers are updated.
0258 1328 :
0258 1329 : CALLING SEQUENCE:
0258 1330 :
0258 1331 : CALLS #1, CLASS_COLLECT
0258 1332 :
0258 1333 : INPUTS:
0258 1334 :
0258 1335 : 4(AP) - address of byte containing class number
0258 1336 :
0258 1337 : IMPLICIT INPUTS:
0258 1338 :
0258 1339 : CDBPTR - pointer to CDB (Class Descriptor Block)
0258 1340 : MRBPTR - pointer to MRB (Monitor Request Block)
0258 1341 : MCAPTR - pointer to MCA (Monitor Communication Area)
0258 1342 : SPTR - pointer to SYI (System Information Area)
0258 1343 :
0258 1344 : OUTPUTS:
0258 1345 :
0258 1346 : None
0258 1347 :
0258 1348 : IMPLICIT OUTPUTS:
0258 1349 :
0258 1350 : Collection buffer filled with raw data for this class
0258 1351 : for this interval.
0258 1352 :
0258 1353 : All required statistics buffers filled with transformed
0258 1354 : data for this class for this interval.
0258 1355 :
0258 1356 : COLLENDED bit set to YES if this collection
0258 1357 : has passed the requested ending time.
0258 1358 :
0258 1359 : ROUTINE VALUE:
0258 1360 :
0258 1361 : R0 = NORMAL
0258 1362 :
0258 1363 : SIDE EFFECTS:
0258 1364 :
0258 1365 : None
0258 1366 :
0258 1367 : REGISTER USAGE:
0258 1368 :
0258 1369 : R6 = CDB pointer
0258 1370 : R7 = MRB pointer
```



```
0258 1371 : R8 = CURRENT collection buffer pointer
0258 1372 : R9 = PREVIOUS collection buffer pointer
0258 1373 : R10 = Buffer block pointer
0258 1374 : R11 = MCA pointer
0258 1375 :
0258 1376 : Others are volatile.
0258 1377 :
0258 1378 :--
0258 1379 :
0258 1380 :
56 00000000'EF OFFC 0258 1381 .ENTRY CLASS_COLLECT, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
57 00000000'EF D0 025A 1382 MOVL CDBPTR,R6 ; Load CDB pointer
5B 00000000'EF D0 0261 1383 MOVL MRBPTR,R7 ; Load MRB pointer
0268 1384 MOVL MCAPTR,R11 ; Load MCA pointer
026F 1385 :
0C AB D5 026F 1386 TSTL MCASL_COLLCNT(R11) ; First collection ?
OC 12 0272 1387 BNEQ 5$ ; No -- keep going
00000000'EF 16 0274 1388 JSB GET_BUFFERS ; Get collection & stat buffers
03 50 E8 027A 1389 BLBS R0,5$ ; Continue if OK
016F 31 027D 1390 BRW CC_ERROR ; Else exit with error
5A 2E A6 D0 0280 1391 5$: MOVL CDB$A_BUFFERS(R6),R10 ; Load address of buffer block
03 4B A6 05 E1 0284 1393 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if not homog class
5A 6A D0 0289 1394 MOVL (R10),R10 ; Get MBP ptr for homog class
028C 1395 :
028C 1396 :
028C 1397 : For standard classes:
028C 1398 :
028C 1399 : Load CURRENT buffer pointer (R8) and PREVIOUS buffer pointer (R9); at
028C 1400 : the same time, flip-flop the state of the CDB$V_SWAPBUF bit, so BUFFERA
028C 1401 : and BUFFERB will be swapped on next CLASS COLLECT call for this class.
028C 1402 : When CDB$V_SWAPBUF is OFF, BUFFERA is CURRENT; otherwise, BUFFERB is
028C 1403 : CURRENT.
028C 1404 :
028C 1405 : For non-standard class (PROCESSES):
028C 1406 :
028C 1407 : CURRENT buffer pointer (R8) always points to BUFFERA. The CDB$V_SWAPBUF
028C 1408 : bit is always clear.
028C 1409 :
028C 1410 :
028C 1411 10$:
18 4B A6 04 E0 028C 1412 BBS #CDB$V_STD,CDB$L_FLAGS(R6),15$ ; Branch if STD class
01 OC AB D1 0291 1413 CMPL MCASL_COLLCNT(R11),#1 ; Second collection coming up?
26 12 0295 1414 BNEQ 20$ ; No -- continue
21 43 A7 02 E1 0297 1415 BBC #MRB$V_SUMMARY,MRB$W_FLAGS(R7),20$ ; Continue if not summarizing
029C 1416 :
029C 1417 : Swap MBP$A_BUFFA and MBP$A_BUFF1ST pointers in order to retain data
029C 1418 : from the first collection buffer for use later during summary processing.
029C 1419 :
029C 1420 :
029C 1421 :
6A 51 6A D0 029C 1421 MOVL MBP$A_BUFFA(R10),R1 ; Save current coll buff ptr
04 04 AA D0 029F 1422 MOVL MBP$A_BUFF1ST(R10),MBP$A_BUFFA(R10) ; Point current to first
04 AA 51 D0 02A3 1423 MOVL R1,MBP$A_BUFF1ST(R10) ; ... and first to current
14 11 02A7 1424 BRB 20$ ; Go make BufferA CURRENT
02A9 1425 :
02A9 1426 : Standard classes
02A9 1427 :
```

					02A9	1428			
					02A9	1429	15\$:		
4B	A6	0F	4B	A6	01	E4	02A9	1430	BBSC
		01		01	01	F0	02AE	1431	INSV
			58		04	AA	D0	02B4	1432
				59		6A	D0	02B8	1433
						07	11	02BB	1434
								02BD	1435
				58		6A	D0	02BD	1436
			59		04	AA	D0	02C0	1437

MOV	MBP\$A_BUFFERB(R10),R8	; Make BufferB CURRENT
MOV	MBP\$A_BUFFERA(R10),R9	; ... and BufferA PREVIOUS
BRB	30\$	; ... and continue

MOV	MBP\$A_BUFFERA(R10),R8	; Make BufferA CURRENT
MOV	MBP\$A_BUFFERB(R10),R9	; ... and BufferB PREVIOUS


```
02C4 1439 :  
02C4 1440 : Collect data for this class into the CURRENT collection buffer.  
02C4 1441 :  
02C4 1442 :  
02C4 1443 30$:  
0439 30 02C4 1444 BSBW COLLECTION : Get data for this class  
03 50 E8 02C7 1445 BLBS R0,40$ : Continue if OK  
0122 31 02CA 1446 BRW CC_ERROR : Else exit with error  
02CD 1447 :  
02CD 1448 : If a record request, perform recording.  
02CD 1449 :  
02CD 1450 :  
02CD 1451 40$:  
44 43 A7 01 E1 02CD 1452 BBC #MRBSV_RECORD,MRBSW_FLAGS(R7),70$ : Continue if not recording  
05 43 A7 03 E1 02D2 1453 BBC #MRBSV_PLAYBACK,MRBSW_FLAGS(R7),50$ : If live, go record  
3A 32 AB 02 E1 02D7 1454 BBC #MCASV_MULTFND,MCASW_FLAGS(R11),70$ : Skip rec if mult not found  
02DC 1455 50$:  
13 4B A6 04 E0 02DC 1456 BBS #CDBSV_STD,CDB$L_FLAGS(R6),60$ : If STD class, go write a record  
7E 20 A6 3C 02E1 1457 MOVZWL CDB$W_BLKLEN(R6),-(SP) : Non-STD class -- push data block size  
58 DD 02E5 1458 PUSHL R8 : ... and collection buffer ptr  
0000048A'EF 02 FB 02E7 1459 CALLS #2,WRITE_PROC_RECORDS : Write the required num of PROCESSES recs  
25 50 E8 02EE 1460 BLBS R0,70$ : Continue if status OK  
00FB 31 02F1 1461 BRW CC_ERROR : Else exit with error  
02F4 1462 60$:  
7E 7C 02F4 1463 CLRQ -(SP) : Get descr on stack for CALL  
6E 20 A6 B0 02F6 1464 MOVW CDB$W_BLKLEN(R6),(SP) : Move in length of buffer  
07 4B A6 05 E1 02FA 1465 BBC #CDBSV_HOMOG,CDB$L_FLAGS(R6),65$ : Br if a heterogeneous class  
6E 0D A8 C4 02FF 1466 MULL2 <MNR_CLSSK_HSIZE+MNR_HOM$ELTCT>(R8),(SP)  
0303 1467 : Times number of elts for homog class  
6E 08 C0 0303 1468 ADDL2 #MNR_HOM$K_PSIZE,(SP) : ... plus the prefix  
0306 1469 65$:  
6E 0D C0 0306 1470 ADDL2 #MNR_CLSSK_HSIZE,(SP) : Add in class header size  
04 AE 58 D0 0309 1471 MOVL R8,47SP) : Load address of buffer  
5E DD 030D 1472 PUSHL SP : Stack descriptor address  
00000000'EF 01 FB 030F 1473 CALLS #1,WRITE_RECORD : ... and record the buffer  
0316 1474 :  
0316 1475 70$:  
00'8F 04 BC 91 0316 1476 CMPB @4(AP),#MODES_CLSNO : Is this the modes class?  
1F 12 031B 1477 BNEQ 80$ : no, branch  
031D 1478 :  
031D 1479 : Combine MODES counters if required  
031D 1480 :  
03 4B A6 03 E1 031D 1481 BBC #CDBSV_CPU_COMB,CDB$L_FLAGS(R6),75$  
00FE 30 0322 1482 : Br if not special MODES case  
0322 1483 BSBW COMBINE_MODES : Combine modes for all cpu's  
0325 1484 :  
0325 1485 :  
0325 1486 :  
0325 1487 : Sum the first six counters to get sum of all CPU modes  
0325 1488 :  
0325 1489 75$:  
00000000'EF D4 0325 1490 CLRL CPU_BUSY : Zero CPU_BUSY  
50 06 D0 032B 1491 MOVL #6,R0 : Get number of modes for display  
51 58 0D C1 032E 1492 ADDL3 #MNR_CLSSK_HSIZE,R8,R1 : Compute start addr of 1st set of counters  
00000000'EF 81 C0 0332 1493 77$:  
F6 50 F5 0332 1494 ADDL2 (R1)+,CPU_BUSY : Sum of non-idle mode counters  
0339 1495 SOBGTR R0,77$ :
```

```
033C 1496
033C 1497
033C 1498 ; Fill STATS buffer for homogeneous class
033C 1499
033C 1500
033C 1501
033C 1502 80$:
7F 4B A6 04 E1 033C 1503 BBC #CDB$V_STD,CDB$L_FLAGS(R6),120$ ; If non-STD class, skip all transfo
0341 1504
11 4B A6 05 E1 0341 1505 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),90$ ; Br if a heterogeneous class
59 DD 0346 1506 PUSHL R9 ; Stack addr of PREV coll buff
58 DD 0348 1507 PUSHL R8 ; Stack addr of CURR coll buff
00000000'EF 02 FB 034A 1508 CALLS #2,FILL_HOMOG_STATS ; Fill STATS buffers for homog class
03 50 E8 0351 1509 BLBS R0,90$ ; Continue if OK
0098 31 0354 1510 BRW CC_ERROR ; Else exit with error
0357 1511 90$:
0357 1512
OC AB D5 0357 1513 TSTL MCAS$L_COLLCNT(R11) ; First collection?
64 13 035A 1514 BEQL 120$ ; Yes -- skip all transforms
035C 1515
035C 1516 ; Calculate MCAS$L_INTTICKS (clock ticks during interval just finished)
035C 1517
035C 1518
035C 1519
52 03 A8 7D 035C 1520 MOVQ MNR_CLSSQ_STAMP(R8),R2 ; Current system time to temp regs
52 03 A9 C2 0360 1521 SUBL2 MNR_CLSSQ_STAMP(R9),R2 ; Calc low-order in sys units
53 07 A9 D9 0364 1522 SBWC MNR_CLSSQ_STAMP+4(R9),R3 ; Calc high-order in sys units
52 08 AB 52 000186A0 8F 7B 0368 1523 EDIV #100000,R2,MCAS$L_INTTICKS(R11),R2 ; Calc interval ticks (10ms units)
0372 1524 ; ... for use later
0372 1525
0372 1526 ; Do Data Transformations for STANDARD (homogeneous) classes
0372 1527
0372 1528
0372 1529
43 4B A6 05 E1 0372 1530 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),110$ ; Br if a heterogeneous class
0377 1531
18 A6 D5 0377 1532 TSTL CDB$L_ECOUNT(R6) ; Any elements in STATS?
44 13 037A 1533 BEQL 120$ ; No -- skip transformations
037C 1534
51 10 D0 037C 1535 MOVL #CDX$S_IBITS,R1 ; Init bit field size
50 D4 037F 1536 CLRL R0 ; Init start position
54 D4 0381 1537 CLRL R4 ; Init item index
0383 1538 100$:
0383 1539
53 52 32 A6 D0 0383 1540 MOVL CDB$A_CDX(R6),R2 ; Get CDB extension for HOMOG class
62 51 50 EA 0387 1541 FFS R0,R1,CDX$W_IBITS(R2),R3 ; Search for next item number
038C 1542 ; R3 contains item number if found
32 13 038C 1543 BEQL 120$ ; Branch if no more items
038E 1544
50 1C B643 9A 038E 1545 MOVZBL @CDB$A_ITMSTR(R6)[R3],R0 ; Load IDB item number
50 11 C4 0393 1546 MULL2 #IDB$K_ILENGTH,R0 ; Compute index into IDB table
50 0000'CF40 9E 0396 1547 MOVAB W^PERFTABLE[R0],R0 ; Address of IDB for this item
00E2'CF 0A A0 B0 039C 1548 MOVW IDB$W_TYPE(R0),W^HOMOG_TYPE ; Save item type for COMPUTE_STATS
03A2 1549
5A 2E B644 D0 03A2 1550 MOVL @CDB$A_BUFFERS(R6)[R4],R10 ; Load MBP ptr for this item
18 BB 03A7 1551 PUSHR #^M<R3,R4> ; Save regs
0044 30 03A9 1552 BSBW TRANSFORMS ; Fill trans'n buffers for this item
```


	18	BA	03AC	1553	POPR	#^M<R3,R4>	; Restore regs
			03AE	1554			
50	53	01	C1	03AE	1555	ADDL3	#1,R3,R0
51	10	50	C3	03B2	1556	SUBL3	R0,#CDX\$\$_IBITS,R1
		54	D6	03B6	1557	INCL	R4
		C9	11	03B8	1558	BRB	100\$
				03BA	1559		
				03BA	1560		
				03BA	1561		; Fill STATS and do Data Transformations for STANDARD (heterogeneous) classes
				03BA	1562		
				03BA	1563		
				03BA	1564		
				03BA	1565		
042C	30	30	03BA	1565	BSBW	FILL_HETERO_STATS	; Fill STATS from the 2 coll buffers
0030	30	30	03BD	1566	BSBW	TRANSFORMS	; ... and fill all transformation buffs

```
03C0 1568 :  
03C0 1569 : Call COLLECTION_END if end of collection has been reached. The end  
03C0 1570 : of collection is tested by comparing the system time (quadword)  
03C0 1571 : values of the current time and the requested end time.  
03C0 1572 :  
03C0 1573 :  
03C0 1574 120$:  
31 AB 04 BC 91 03C0 1575 CMPB @4(AP),MCASB_LASTC(R11) ; Last class?  
21 12 03C5 1576 BNEQU CC_NORMAL ; No -- just exit  
28 AB 03 A8 7D 03C7 1577 MOVQ MNR_CLSSQ_STAMP(R8),MCASQ_LASTCOLL(R11) ; Yes -- remember latest col  
17 43 A7 04 E0 03CC 1578 BBS #MRBSV_INDEFEND,MRBSW_FLAGS(R7),CC_NORMAL ; Skip end check if indef  
0C A7 07 A8 D1 03D1 1579 CMPL MNR_CLSSQ_STAMP+4(R8),MRBSQ_ENDING+4(R7) ; Has curr time passed  
10 1F 03D6 1580 ; ... requested end (hi-orde  
07 1A 03D8 1581 BLSSU CC_NORMAL ; No -- simply return  
08 A7 03 A8 D1 03DA 1582 BGTRU 130$ ; Yes -- indicate so and ret  
07 1F 03DF 1583 CMPL MNR_CLSSQ_STAMP(R8),MRBSQ_ENDING(R7) ; Check low order longword  
03E1 1584 BLSSU CC_NORMAL ; Not at end yet -- return  
00000000'EF 00 FB 03E1 1585 130$: CALLS #0,COLLECTION_END ; Indicate collection has en  
03E8 1586 ; COLLECTION_END sets COLLEN  
03E8 1587  
03E8 1588  
50 00000000'EF D0 03E8 1589 CC_NORMAL: MOVL NORMAL,R0 ; Indicate normal status  
03EF 1590 CC_ERROR: ;  
04 03EF 1591 RET ; Return  
03EF 1592
```



```
03F0 1594      .SBTTL TRANSFORMS - Perform Data Transformations
03F0 1595
03F0 1596      :++
03F0 1597      :
03F0 1598      : FUNCTIONAL DESCRIPTION:
03F0 1599      :
03F0 1600      : This routine updates all transformation buffers (STATS, MIN,
03F0 1601      : MAX, SUM, PCSTATS, PCMIN, PCMAX, PCSUM) for STANDARD classes
03F0 1602      : (both homogeneous and heterogeneous).
03F0 1603      :
03F0 1604      : CALLING SEQUENCE:
03F0 1605      :
03F0 1606      :     BSBW TRANSFORMS
03F0 1607      :
03F0 1608      : INPUTS:
03F0 1609      :
03F0 1610      :     R6 - CDB pointer
03F0 1611      :     R7 - MRB pointer
03F0 1612      :     R8 - CURRENT buffer pointer
03F0 1613      :     R9 - PREVIOUS buffer pointer
03F0 1614      :     R10 - Buffer block (MBP) pointer
03F0 1615      :     R11 - MCA pointer
03F0 1616      :
03F0 1617      : IMPLICIT INPUTS:
03F0 1618      :
03F0 1619      :     All transformation buffers
03F0 1620      :
03F0 1621      : OUTPUTS:
03F0 1622      :
03F0 1623      :     None
03F0 1624      :
03F0 1625      : IMPLICIT OUTPUTS:
03F0 1626      :
03F0 1627      :     All transformation buffers updated with statistics
03F0 1628      :     from the current collection.
03F0 1629      :
03F0 1630      : ROUTINE VALUE:
03F0 1631      :
03F0 1632      :     None
03F0 1633      :
03F0 1634      : SIDE EFFECTS:
03F0 1635      :
03F0 1636      :     Registers R0,R1,R2,R3,R4,R5 destroyed.
03F0 1637      :
03F0 1638      :--
03F0 1639      :
03F0 1640      : TRANSFORMS:                                ; Perform data transformations
03F0 1641      :
03F0 1642      : ; Update SUM buffer from STATS buffer
03F0 1643      :
03F0 1644      :
03F0 1645      :
50 08 AA DO 03F0 1646      MOVL MBP$A_STATS(R10),R0      ; Load STATS buffer pointer
51 14 AA DO 03F4 1647      MOVL MBP$A_SUM(R10),R1        ; Load SUM buffer pointer
52 18 A6 DO 03F8 1648      MOVL CDB$E_COUNT(R6),R2        ; Load count of elements in STATS
      03FC 1649      10$:
      81 80 C0 03FC 1650      ADDL2 (R0)+,(R1)+            ; Add this item to SUM buff
```

```
FA 52 F5 03FF 1651 SOBGTR R2,10$ ; Loop for each item in STATS buff
                                0402 1652
18 45 A6 00 E1 0402 1653 BBC #CDB$V_PERCENT,CDB$W_QFLAGS(R6),30$ ; If percent not requested, skip
                                0407 1654
049E 30 0407 1655 BSBW FILL_PCSTATS_BUFF ; Fill PCSTATS from STATS
                                040A 1656
                                040A 1657 ;
                                040A 1658 ; Update PCSUM buffer from PCSTATS buffer
                                040A 1659 ;
50 18 AA D0 040A 1660 MOVL MBP$A_PCSTATS(R10),R0 ; Load PCSTATS buffer pointer
51 24 AA D0 040E 1661 MOVL MBP$A_PCSUM(R10),R1 ; Load PCSUM buffer pointer
52 18 A6 D0 0412 1662 MOVL CDB$$_ECOUNT(R6),R2 ; Load count of elements in PCSTATS
                                0416 1663 20$:
81 80 C0 0416 1664 ADDL2 (R0)+(R1)+ ; Add this item to PCSUM buff
FA 52 F5 0419 1665 SOBGTR R2,20$ ; Loop for each item in PCSTATS buff
                                041C 1666 ;
                                041C 1667 ; Update PCMIN and PCMAX buffers from PCSTATS buffer
                                041C 1668 ;
0552 30 041C 1669 BSBW UPD_PC_MIN_MAX ; Update PCMIN and PCMAX
                                041F 1670
                                041F 1671 ;
                                041F 1672 ; Convert counts to rates in STATS buffer and update MIN and MAX
                                041F 1673 ;
                                041F 1674
04BD 30 041F 1675 30$:
                                0422 1676 BSBW COMPUTE_STATS ; Convert counts to rates in STATS
                                0422 1677 ; ... and update MIN and MAX
                                0422 1678
05 0422 1679 RSB ; Return from TRANSFORMS subroutine
```



```
0423 1681      .SBTTL COMBINE_MODES - Combine Modes for all CPUs
0423 1682
0423 1683 :++
0423 1684 :
0423 1685 : FUNCTIONAL DESCRIPTION:
0423 1686 :
0423 1687 :     This routine is called by CLASS_COLLECT to combine the mode
0423 1688 :     tick counters for all CPU's on the system. The monitored
0423 1689 :     system is a multiprocessing system, but the user requested
0423 1690 :     that the display and/or summary show combined values for
0423 1691 :     the processor modes (/NOCPU).
0423 1692 :
0423 1693 : CALLING SEQUENCE:
0423 1694 :
0423 1695 :     BSBW    COMBINE_MODES
0423 1696 :
0423 1697 : INPUTS:
0423 1698 :
0423 1699 :     None
0423 1700 :
0423 1701 : IMPLICIT INPUTS:
0423 1702 :
0423 1703 :     R6 = Pointer to MODES CDB
0423 1704 :     R8 = Pointer to CURRENT collection buffer
0423 1705 :
0423 1706 : OUTPUTS:
0423 1707 :
0423 1708 :     None
0423 1709 :
0423 1710 : IMPLICIT OUTPUTS:
0423 1711 :
0423 1712 :     The first 7 longwords in the data portion of the collection buffer
0423 1713 :     will contain combined mode counter values for all CPU's on the system.
0423 1714 :
0423 1715 : ROUTINE VALUE:
0423 1716 :
0423 1717 :     None
0423 1718 :
0423 1719 : SIDE EFFECTS:
0423 1720 :
0423 1721 :     Registers R0, R1, R2 destroyed.
0423 1722 :
0423 1723 :--
0423 1724 :
0423 1725 :
0423 1726 COMBINE_MODES:
0423 1727
50 18 A6 D0 0423 1728      MOVL    CDB$E_COUNT(R6),R0      ; Get number of modes for display
51 58 0D C1 0427 1729      ADDL3   #MNR_CLSSK_HSIZE,R8,R1    ; Compute start addr of 1st set of counters
52 6140 DE 042B 1730      MOVAL    (R1)[R0],R2      ; Compute start addr of 2nd set of counters
042F 1731 10$:
81 82 C0 042F 1732      ADDL2   (R2)+,(R1)+      ; Combine 2nd set with 1st set
FA 50 F5 0432 1733      SOBGTR   R0,10$      ; ... for all counters
0435 1734
05 0435 1735      RSB      ; Return
```

```
0436 1737 .SBTTL QUAD_LT_QUAD - Compare Two Quadwords
0436 1738
0436 1739 :++
0436 1740
0436 1741 : FUNCTIONAL DESCRIPTION:
0436 1742
0436 1743 : This routine is called by PL/I routines to compare two unsigned
0436 1744 : quadword values (such as system time values). The routine answers
0436 1745 : the question: Is the first value less than the second value?
0436 1746 : The value YES or NO is placed in R0 upon exit.
0436 1747
0436 1748 : CALLING SEQUENCE:
0436 1749
0436 1750 : CALLS #2,QUAD_LT_QUAD
0436 1751
0436 1752 : INPUTS:
0436 1753
0436 1754 : 4(AP) - address of first quadword value
0436 1755
0436 1756 : 8(AP) - address of second quadword value
0436 1757
0436 1758 : IMPLICIT INPUTS:
0436 1759
0436 1760 : None
0436 1761
0436 1762 : OUTPUTS:
0436 1763
0436 1764 : Routine value below.
0436 1765
0436 1766 : IMPLICIT OUTPUTS:
0436 1767
0436 1768 : None
0436 1769
0436 1770 : ROUTINE VALUE:
0436 1771
0436 1772 : R0 = YES if first quadword value is less than the second quadword value.
0436 1773 : R0 = NO otherwise
0436 1774
0436 1775 : SIDE EFFECTS:
0436 1776
0436 1777 : None
0436 1778
0436 1779 :--
0436 1780
0436 1781
0436 1782 .ENTRY QUAD_LT_QUAD, ^M<>
0438 1783
0438 1784 MOVQ 4(AP),R0
043C 1785 CMPL 4(R0),4(R1)
0441 1786 BLSSU 20$
0443 1787 BGTRU 10$
0445 1788 CMPL (R0),(R1)
0448 1789 BLSSU 20$
044A 1790 10$:
044A 1791 MOVL #NO,R0
044D 1792 BRB QLQ_RET
044F 1793 20$:

0000
50 04 AC 7D 0438 1784 MOVQ 4(AP),R0 ; Pointers in R0 and R1
04 A1 04 A0 D1 043C 1785 CMPL 4(R0),4(R1) ; First < Second (hi-order) ?
OC 1F 0441 1786 BLSSU 20$ ; Yes -- answer is YES
05 1A 0443 1787 BGTRU 10$ ; No -- answer is NO
61 60 D1 0445 1788 CMPL (R0),(R1) ; Check low order longword
05 1F 0448 1789 BLSSU 20$ ; Go answer YES
044A 1790 10$:
50 00 D0 044A 1791 MOVL #NO,R0 ; First not less than second
03 11 044D 1792 BRB QLQ_RET ; Go return
044F 1793 20$:
```


MONITOR
V04-000

- VAX/VMS Performance Monitor Utility L 15
QUAD_LT_QUAD - Compare Two Quadwords

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 45
(22)

50 01 D0 044F 1794 MOVL #YES,R0
04 0452 1795 QLQ_RET:
04 0452 1796 RET

; First less than second
; Return with value in R0

```
0453 1798 .SBTTL QUAD_EQ_0 - Compare Quadword = 0
0453 1799
0453 1800 :++
0453 1801 :
0453 1802 : FUNCTIONAL DESCRIPTION:
0453 1803 :
0453 1804 : This routine is called by PL/I routines to compare an unsigned
0453 1805 : quadword values (such as a system time value) with the quadword
0453 1806 : value 0. The routine answers the question: Is the quadword value
0453 1807 : equal to 0?
0453 1808 :
0453 1809 : The value YES or NO is placed in R0 upon exit.
0453 1810 :
0453 1811 : CALLING SEQUENCE:
0453 1812 :
0453 1813 : CALLS #1,QUAD_EQ_0
0453 1814 :
0453 1815 : INPUTS:
0453 1816 :
0453 1817 : 4(AP) - address of quadword value
0453 1818 :
0453 1819 : IMPLICIT INPUTS:
0453 1820 :
0453 1821 : None
0453 1822 :
0453 1823 : OUTPUTS:
0453 1824 :
0453 1825 : Routine value below.
0453 1826 :
0453 1827 : IMPLICIT OUTPUTS:
0453 1828 :
0453 1829 : None
0453 1830 :
0453 1831 : ROUTINE VALUE:
0453 1832 :
0453 1833 : R0 = YES if quadword value is equal to 0.
0453 1834 : R0 = NO otherwise
0453 1835 :
0453 1836 : SIDE EFFECTS:
0453 1837 :
0453 1838 : None
0453 1839 :
0453 1840 :--
0453 1841 :
0453 1842 :
0000 0453 1843 .ENTRY QUAD_EQ_0, ^M<>
0453 1844 :
50 04 BC 7D 0455 1845 MOVQ @4(AP),R0 ; Quadword value in R0, R1
50 50 D5 0459 1846 TSTL R0 ; Right half = 0?
09 12 045B 1847 BNEQU 10$ ; No -- answer NO
51 D5 045D 1848 TSTL R1 ; Yes -- left half = 0?
05 12 045F 1849 BNEQU 10$ ; No -- answer NO
50 01 D0 0461 1850 MOVL #YES,R0 ; Answer YES
03 11 0464 1851 BRB QEZ_RET ; Go exit
0466 1852 10$:
50 00 D0 0466 1853 MOVL #NO,R0 ; Answer NO
0469 1854 QEZ_RET:
```


- VAX/VMS Performance Monitor Utility N.15
QUAD_EQ_0 - Compare Quadword = 0

Page 47
(23)

```
; Return with value in R0
```

```
046A 1857      .SBTTL MPCHECK - Check system for MP capability
046A 1858
046A 1859      :++
046A 1860      :
046A 1861      : FUNCTIONAL DESCRIPTION:
046A 1862      :
046A 1863      : This routine is called by the EXECUTE_REQUEST PL/I routine
046A 1864      : to determine whether or not the running system has MP
046A 1865      : (multiprocessing) capability.
046A 1866      :
046A 1867      : The value YES or NO is placed in R0 upon exit.
046A 1868      :
046A 1869      : CALLING SEQUENCE:
046A 1870      :
046A 1871      : CALLS #0,QUAD_EQ_0
046A 1872      :
046A 1873      : INPUTS:
046A 1874      :
046A 1875      : None
046A 1876      :
046A 1877      : IMPLICIT INPUTS:
046A 1878      :
046A 1879      : EXE$GB_CPUTYPE -- CPU type. Assume type 1 = 780.
046A 1880      :
046A 1881      : EXE$GL_RPB -- Address of Restart Parameter Block.
046A 1882      :
046A 1883      : OUTPUTS:
046A 1884      :
046A 1885      : Routine value below.
046A 1886      :
046A 1887      : IMPLICIT OUTPUTS:
046A 1888      :
046A 1889      : None
046A 1890      :
046A 1891      : ROUTINE VALUE:
046A 1892      :
046A 1893      : R0 = YES if the system has MP capability
046A 1894      : R0 = NO otherwise
046A 1895      :
046A 1896      : SIDE EFFECTS:
046A 1897      :
046A 1898      : None
046A 1899      :
046A 1900      :--
046A 1901      :
046A 1902      :
0000 046A 1903 .ENTRY MPCHECK,      ^M<>
046C 1904
01  00000000'GF  91 046C 1905      CMPB      G^EXE$GB_CPUTYPE,#1      ; 780 processor?
11  12 0473 1906      BNEQ      10$      ; No -- go answer NO
50  00000000'GF  D0 0475 1907      MOVL      G^EXE$GL_RPB,R0      ; Get Restart Parameter Block ptr
05  30 A0 0B E1 047C 1908      BBC      #RPB$V_MPM,RPB$L_BOOTR5(R0),10$ ; Br if no multi-port mem
50  01 D0 0481 1909      MOVL      #YES,R0      ; Answer YES
03  11 0484 1910      BRB      20$      ; Go exit
50  00 D0 0486 1911 10$:      MOVL      #NO,R0      ; Answer NO
0486 1912
0489 1913 20$:
```


MONITOR
V04-000

- VAX/VMS Performance Monitor Utility C 16
MPCHECK - Check system for MP capability 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 49
(24)

04 0489 1914

RET

; Return with value in R0

```
048A 1916 .SBTTL WRITE_PROC_RECORDS - Write PROCESSES class records
048A 1917
048A 1918 :++
048A 1919 :
048A 1920 : FUNCTIONAL DESCRIPTION:
048A 1921 :
048A 1922 : This routine is called to write out a group of class
048A 1923 : records containing the data in a PROCESSES collection
048A 1924 : buffer.
048A 1925 :
048A 1926 : CALLING SEQUENCE:
048A 1927 :
048A 1928 : CALLS #2,WRITE_PROCS_RECORDS
048A 1929 :
048A 1930 : INPUTS:
048A 1931 :
048A 1932 : 4(AP) - address of collection buffer
048A 1933 :
048A 1934 : 8(AP) - longword size of a data block (for a single process)
048A 1935 :
048A 1936 : IMPLICIT INPUTS:
048A 1937 :
048A 1938 : None
048A 1939 :
048A 1940 : OUTPUTS:
048A 1941 :
048A 1942 : None
048A 1943 :
048A 1944 : IMPLICIT OUTPUTS:
048A 1945 :
048A 1946 : As many class records as are required are written to the
048A 1947 : recording file for this PROCESSES collection buffer.
048A 1948 :
048A 1949 : ROUTINE VALUE:
048A 1950 :
048A 1951 : R0 = NORMAL, or error status, if any
048A 1952 :
048A 1953 : SIDE EFFECTS:
048A 1954 :
048A 1955 : None
048A 1956 :
048A 1957 :--
048A 1958 :
048A 1959 :
OFFC 048A 1960 .ENTRY WRITE_PROC_RECORDS, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
048C 1961
57 04 AC D0 048C 1962 MOVL 4(AP),R7 ; Get pointer to collection buffer
58 11 A7 D0 0490 1963 MOVL <MNR_CLSSK_HSIZE+MNR_PROSL_PCTINT>(R7),R8
0494 1964 ; Get num of processes in coll buffer
58 0016'CF D1 0494 1965 CMPL W^PROCS_PER_REC,R8 ; Need more than 1 class record?
25 19 0499 1966 BLSS 10$ ; Yes -- go break it up
049B 1967 ALLOC 8,R1,R2 ; No -- alloc a descriptor on stack
62 08 AC 58 C5 04A8 1968 MULL3 R8,8(AP),(R2) ; Calculate size of data area
62 62 15 C0 04AD 1969 ADDL2 #<MNR_CLSSK_HSIZE+MNR_PROSL_PSIZE>(R2)
04B0 1970 ; Add in class hdr and PROCESSES prefix
04B0 1971 MOVL R7,4(R2) ; Load addr of coll buff into descr
04B4 1972 PUSHL R2 ; Push descriptor address
```


MONITOR
V04-000

E 16
- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00 Page 51
WRITE_PROC_RECORDS - Write PROCESSES cla 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 (25)

00000000'EF 01 FB 04B6 1973 CALLS #1 WRITE_RECORD ; ... and write the record
00C6 31 04BD 1974 BRW WPR_RET ; Go return with status of WRITE_RECORD

```
04C0 1976 :  
04C0 1977 : Break up collection buffer into a group of records and write  
04C0 1978 : each into the recording file. All except the last will have the  
04C0 1979 : MNR_CLSSV_CONT flag set indicating that data for this interval  
04C0 1980 : continues in the next record.  
04C0 1981 :  
04C0 1982 :  
04C0 1983 : Register Usage:  
04C0 1984 :  
04C0 1985 : R6 = # of full records to write  
04C0 1986 : R7 = collection buffer index pointer  
04C0 1987 : R8 = # of processes in the collection buffer;  
04C0 1988 : also, # of processes in the final record  
04C0 1989 : R9 = pointer to write buffer data area  
04C0 1990 : R10 = size of data portion of write buffer  
04C0 1991 : R11 = pointer to write buffer descriptor  
04C0 1992 :  
04C0 1993 :  
04C0 1994 10$:  
56 58 0016'CF C7 04C0 1995 DIVL3 W^PROCS_PER_REC,R8,R6 : Get number of full records to write  
51 56 0016'CF C5 04C6 1996 MULL3 W^PROCS_PER_REC,R6,R1 : Calculate # of procs in ...  
58 51 C2 04CC 1997 SUBL2 R1,R8 : ... final record  
SA 08 AC 0016'CF C5 04CF 1998 MULL3 W^PROCS_PER_REC,8(AP),R10 : Get size of data portion of write buff  
51 5A 15 C1 04D6 1999 ADDL3 #<MNR_CLSSK_HSIZE+MNR_PROSK_PSIZE>,R10,R1 :  
04DA 2000 : Compute write buffer size  
04DA 2001 ALLOC 8,R0,R11 : Get a write buffer descr on stack  
6B 51 D0 04E7 2002 MOVL R1,(R11) : Move in write buff size  
04 AB DF 04EA 2003 PUSHAL 4(R11) : Push addr of write buffer ptr  
6B DF 04ED 2004 PUSHAL (R11) : Push addr of write buffer size  
00000000'GF 02 FB 04EF 2005 CALLS #2,G^LIB$GET_VM : Get the write buffer  
03 50 E8 04F6 2006 BLBS R0,20$ : Continue if status OK  
008A 31 04F9 2007 BRW WPR_RET : Return with status if failed  
04FC 2008 20$:  
001A'CF 6B 7D 04FC 2009 MOVQ (R11),W^PROC_WRI_BUFD : Save descriptor for later cleanup  
59 04 AB D0 0501 2010 MOVL 4(R11),R9 : Get ptr to write buffer  
69 67 15 28 0505 2011 MOVCL3 #<MNR_CLSSK_HSIZE+MNR_PROSK_PSIZE>,(R7),(R9) :  
0509 2012 : Move class hdr & prefix to write buff  
0509 2013 ADDL2 #<MNR_CLSSK_HSIZE+MNR_PROSK_PSIZE>,R7 : Update coll buff ptr  
00 01 A9 00 E2 050C 2014 BBSS #MNR_CLSSV_CONT,MNR_CLSSW_FLAGS(R9),30$ : Set 'continued' bit  
59 0D C0 0511 2015 ADDL2 #MNR_CLSSK_HSIZE,R9 : Point to PROCESSES prefix  
69 0016'CF D0 0514 2016 MOVL W^PROCS_PER_REC,MNR_PROSL_PCTREC(R9) : Load # of procs this rec  
59 08 C0 0519 2017 ADDL2 #MNR_PROSK_PSIZE,R9 : Point to data portion of write buffer
```



```
051C 2019 :  
051C 2020 : Loop until all processes which fit into full records  
051C 2021 : have been handled. On each time through the loop, move  
051C 2022 : PROCS_PER_REC processes from the collection buffer to  
051C 2023 : the write buffer and write it out. The class header  
051C 2024 : portion and the PROCESSES prefix portion of the write  
051C 2025 : buffer will be identical for all these records.  
051C 2026 :  
051C 2027 :  
051C 2028 40$:  
69 67 5A 28 051C 2029 MOV C3 R10,(R7),(R9) ; Move group of procs from coll to write  
57 5A C0 0520 2030 ADD L2 R10,R7 ; Update coll buff ptr  
00000000'EF 5B DD 0523 2031 PUSH L R11 ; Stack ptr to write buffer descr  
57 01 FB 0525 2032 CALLS #1,WRITE_RECORD ; Write this record  
EA 50 E9 052C 2033 BLBC R0,WPR_RET ; Go exit with status if failed  
56 F5 052F 2034 SOB GTR R6,40$ ; Loop back to do next group  
0532 2035 :  
0532 2036 : Build and write a final record for the "leftover"  
0532 2037 : processes which didn't fit into one of the full records.  
0532 2038 :  
0532 2039 :  
51 04 AB D0 0532 2040 MOVL 4(R11),R1 ; Get pointer to write buff class hdr  
00 01 A1 00 E5 0536 2041 BBCC #MNR_CLSSV_CONT,MNR_CLSSW_FLAGS(R1),50$ ; Clear "continued" bit  
51 0D C0 053B 2042 50$: ADD L2 #MNR_CLSSK_HSIZE,R1 ; Point to PROCESSES prefix  
61 58 D0 053E 2043 MOVL R8,MNR_PROSL_PCTREC(R1) ; Load # of procs in this record  
5A 58 08 AC C5 0541 2044 MULL3 8(AP),R8,R10 ; Compute size of data portion  
5A D5 0546 2045 TSTL R10 ; Any "leftover" data to write?  
69 67 04 13 0548 2046 BEQL 60$ ; No -- skip around the MOV C3  
5A 28 054A 2047 MOV C3 R10,(R7),(R9) ; Move leftovers from coll to write buff  
054E 2048 60$:  
04 A1 04 AB D0 055B 2049 ALLOC 8,R0,R1 ; Allocate a descr for the write  
61 5A 15 C1 0560 2050 MOVL 4(R11),4(R1) ; Get write buff ptr from previous descr  
0564 2051 ADD L3 #<MNR_CLSSK_HSIZE+MNR_PROSK_PSIZE>,R10,(R1)  
0564 2052 ; Move write buff length into descr  
00000000'EF 51 DD 0564 2053 PUSH L R1 ; Stack descr ptr for write  
16 50 E9 0566 2054 CALLS #1,WRITE_RECORD ; Write the last record  
056D 2055 BLBC R0,WPR_RET ; Leave with status if failed  
0570 2056 :  
0570 2057 : Free the virtual memory occupied by the write buffer  
0570 2058 :  
0570 2059 :  
04 AB DF 0570 2060 PUSHAL 4(R11) ; Stack addr of write buffer ptr  
00000000'GF 6B DF 0573 2061 PUSHAL (R11) ; Stack addr of write buffer len  
07 02 FB 0575 2062 CALLS #2,G^LIB$FREE_VM ; Free the write buffer memory  
50 00000000'EF 07 50 E9 057C 2063 BLBC R0,WPR_RET ; Leave with status if failed  
D0 057F 2064 MOVL NORMAL,R0 ; Indicate normal status  
0586 2065 WPR_RET:  
04 0586 2066 RET ; Return with status
```

```
0587 2068 .SBTTL CVT_TO_DELTA - Convert Seconds to Delta
0587 2069
0587 2070 :++
0587 2071 :
0587 2072 : FUNCTIONAL DESCRIPTION:
0587 2073 :
0587 2074 : This routine is called to convert a positive seconds quantity
0587 2075 : supplied in a longword to a quadword delta time quantity.
0587 2076 :
0587 2077 : CALLING SEQUENCE:
0587 2078 :
0587 2079 : CALLS #2,CVT_TO_DELTA
0587 2080 :
0587 2081 : INPUTS:
0587 2082 :
0587 2083 : 4(AP) - address of longword containing positive seconds quantity
0587 2084 :
0587 2085 : 8(AP) - address of quadword in which to store converted delta time.
0587 2086 :
0587 2087 : IMPLICIT INPUTS:
0587 2088 :
0587 2089 : None
0587 2090 :
0587 2091 : OUTPUTS:
0587 2092 :
0587 2093 : Quadword addressed by 8(AP) is loaded with converted delta time.
0587 2094 :
0587 2095 : IMPLICIT OUTPUTS:
0587 2096 :
0587 2097 : None
0587 2098 :
0587 2099 : ROUTINE VALUE:
0587 2100 :
0587 2101 : None
0587 2102 :
0587 2103 : SIDE EFFECTS:
0587 2104 :
0587 2105 : None
0587 2106 :
0587 2107 :--
0587 2108 :
0587 2109 :
0000 0587 2110 .ENTRY CVT_TO_DELTA, ^M<>
0589 2111
00 04 BC FF676980 8F 7A 0589 2112 EMUL #-10*1000*1000,@4(AP),#0,@8(AP) ; That's all, folks
08 BC 0592
04 0594 2113 RET
```



```
0595 2115 .SBTTL COMPUTE_BOOTTIME - Compute System Time of Boot
0595 2116
0595 2117 :++
0595 2118
0595 2119 : FUNCTIONAL DESCRIPTION:
0595 2120
0595 2121 : This routine is called to compute the quadword time value
0595 2122 : representing the absolute time at which the monitored
0595 2123 : (running) system was booted. This is done by converting
0595 2124 : the EXESGL_ABSTIM value (absolute number of seconds since
0595 2125 : boot) to a negative quadword system time value and adding
0595 2126 : it to the current time (obtained via $GETTIM).
0595 2127
0595 2128 : CALLING SEQUENCE:
0595 2129
0595 2130 : CALLS #0,COMPUTE_BOOTTIME
0595 2131
0595 2132 : INPUTS:
0595 2133
0595 2134 : None
0595 2135
0595 2136 : IMPLICIT INPUTS:
0595 2137
0595 2138 : EXESGL_ABSTIM - longword containing positive number of seconds
0595 2139 : since boot.
0595 2140
0595 2141 : SPTR - pointer to SYI (System Information Area).
0595 2142
0595 2143 : OUTPUTS:
0595 2144
0595 2145 : None
0595 2146
0595 2147 : IMPLICIT OUTPUTS:
0595 2148
0595 2149 : MNR_SYISQ_BOOTTIME loaded with boot time in system time units.
0595 2150
0595 2151 : ROUTINE VALUE:
0595 2152
0595 2153 : R0 = NORMAL, or failing system service status
0595 2154
0595 2155 : SIDE EFFECTS:
0595 2156
0595 2157 : None
0595 2158
0595 2159 :--
0595 2160
0004 0595 2161
0595 2162 .ENTRY COMPUTE_BOOTTIME, ^M<R2>
0597 2163
0597 2164 ALLOC 8,R0,R2 ; Get quadword on stack
05A4 2165 $GETTIM,S,TIMADR=(R2) ; Put current time into it
05AD 2166 BLBC R0,10$ ; Exit if error
0580 2167 EMUL #-10*1000*1000,EXESGL_ABSIM,#0,R0
058C
058D 2168 ; Get delta quadword system time units
058D 2169 ; ... since boot
058D 2170 ADDL2 (R2),R0 ; Add low-order current time
```

```
00 00000000'EF 26 50 E9
FF676980 8F 7A
50
50 62 C0
```

MONITOR
V04-000

J 16
- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00 Page 56
COMPUTE_BOOTTIME - Compute System Time o 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 (29)

```
52      51      04 A2      D8 05C0 2171      ADWC 4(R2),R1      ; ... and high-order current time
      00000000'EF      DO 05C4 2172      MOVL SPTR,R2      ; Get pointer to SYI
      03 A2      50      7D 05CB 2173      MOVQ R0,MNR_SYISQ_BOOTTIME(R2) ; Move in boot time
50      00000000'EF      DO 05CF 2174      MOVL NORMAL,R0      ; Indicate success
      05D6 2175 10$:      RET      ; Return
      04 05D6 2176
```



```
05D7 2178 .SBTTL CLUS_NET_INFO - Get Cluster & Net Info
05D7 2179
05D7 2180 :++
05D7 2181 :
05D7 2182 : FUNCTIONAL DESCRIPTION:
05D7 2183 :
05D7 2184 : This routine is called to gather certain info about the
05D7 2185 : cluster and/or network and place it into the System
05D7 2186 : Information Area.
05D7 2187 :
05D7 2188 : CALLING SEQUENCE:
05D7 2189 :
05D7 2190 : CALLS #0, CLUS_NET_INFO
05D7 2191 :
05D7 2192 : INPUTS:
05D7 2193 :
05D7 2194 : None
05D7 2195 :
05D7 2196 : IMPLICIT INPUTS:
05D7 2197 :
05D7 2198 : CLUS$GL_CLUB - If non-zero, then this node is a cluster member;
05D7 2199 : otherwise, it is not.
05D7 2200 :
05D7 2201 : SPTR - pointer to SYI (System Information Area).
05D7 2202 :
05D7 2203 : SCSNODE system parameter -- SCS node name in ASCII
05D7 2204 : SYSSNODE logical name -- DECnet node name in ASCII
05D7 2205 :
05D7 2206 : OUTPUTS:
05D7 2207 :
05D7 2208 : None
05D7 2209 :
05D7 2210 : IMPLICIT OUTPUTS:
05D7 2211 :
05D7 2212 : SYI updated.
05D7 2213 :
05D7 2214 : ROUTINE VALUE:
05D7 2215 :
05D7 2216 : RO = NORMAL, or failing system service status
05D7 2217 :
05D7 2218 : SIDE EFFECTS:
05D7 2219 :
05D7 2220 : None
05D7 2221 :
05D7 2222 : --
05D7 2223 :
05D7 2224 :
007C 05D7 2225 .ENTRY CLUS_NET_INFO, ^M<R2,R3,R4,R5,R6>
05D9 2226
56 00000000'EF D0 05D9 2227 MOVL SPTR,R6 ; Get System Info Area pointer
05E0 2228
05E0 2229 :
05E0 2230 : Set MNR_SYISV_CLUSMEM if this node is a cluster member
05E0 2231 :
05E0 2232 :
00 01 A6 00 E5 05E0 2233 IFCLSTR 20$ ; Br if a cluster member
05E8 2234 BBCC #MNR_SYISV_CLUSMEM, - ; Else indicate not member
```

```
00 01 A6 05 11 05ED 2235 MNR_SYISW_FLAGS(R6),10$
00 01 A6 00 E2 05ED 2236 10$: BRB 30$-
05EF 2237 20$: BBSS #MNR_SYISV CLUSMEM, - ; ... and go continue
05F4 2238 MNR_SYISW_FLAGS(R6),30$ ; Indicate a cluster member
05F4 2239 30$:
05F4 2240
05F4 2241
05F4 2242 : Call $GETSYI to get CPU type and node name (from SCSNODE parameter)
05F4 2243 :
05F4 2244
05F4 2245 ALLOC 56,R0,R2 ; Allocate local temp storage
0601 2246
0601 2247 :
0601 2248 : Set up item descriptor for node name
0601 2249 :
0601 2250
02 A2 62 10 B0 0601 2251 MOVW #16,(R2) ; Move in length of buffer
04 A2 10D9 8F B0 0604 2252 MOVW #SYIS_NODENAME,2(R2) ; Move in item identifier
08 A2 1C A2 DE 060A 2253 MOVAL 28(R2),4(R2) ; Move in ptr to buffer
08 A2 2C A2 DE 060F 2254 MOVAL 44(R2),8(R2) ; Move in ptr to length word
0614 2255
0614 2256 :
0614 2257 : Set up item descriptor for CPU type
0614 2258 :
0614 2259
0E A2 0C A2 04 B0 0614 2260 MOVW #4,12(R2) ; Move in length of buffer
10 A2 2000 8F B0 0618 2261 MOVW #SYIS_CPU,14(R2) ; Move in item identifier
14 A2 30 A2 DE 061E 2262 MOVAL 48(R2),16(R2) ; Move in ptr to buffer
14 A2 34 A2 DE 0623 2263 MOVAL 52(R2),20(R2) ; Move in ptr to length word
18 A2 18 A2 D4 0628 2264 CLRL 24(R2) ; Indicate end of item list
062B 2265 $GETSYIW S ITMLST=(R2) ; Get the CPU type & SCSNODE node name
03 50 E8 063E 2266 BLBS R0,40$ ; Continue if status OK
0082 31 0641 2267 BRW CNI_RET ; Go exit if error
26 A6 30 A2 D0 0644 2268 40$: MOVL 48(R2),MNR_SYISL_CPUYPE(R6) ; Pick up CPU type
0649 2269
0649 2270 :
0649 2271 : Now process node name
0649 2272 :
0649 2273
53 04 A2 D0 0649 2274 MOVL 4(R2),R3 ; Set up ptr to 1st byte of string
54 08 A2 D0 064D 2275 MOVL 8(R2),R4 ; Set up ptr to length word
64 B5 0651 2276 TSTW (R4) ; Check for null node name
43 12 0653 2277 BNEQ 70$ ; Br if not null
0655 2278
0655 2279 :
0655 2280 : $GETSYI returned a null node name, so try to get the
0655 2281 : DECnet node name by translating logical name SYSSNODE.
0655 2282 :
0655 2283
0655 2284 ALLOC 15,R2,R3 ; Get the result buffer on stack
0662 2285 ALLOC 2,R0,R4 ; ... and a word for actual length
066F 2286
066F 2287 $TRNLOG_S LOGNAM=W^SYSNOD_NAM, RSLBUF=(R2), -
066F 2288 RSLLEN=(R4), DS8MSK=#^B110
0684 2289
0684 2290
3F 50 E9 0684 2291 BLBC R0,CNI_RET ; Translate SYSSNODE in system table
; Exit if error
```



```
50 00000629 8F D1 0687 2292
      08 12 0687 2293      CMPL #SS$_NOTRAN,R0      ; SYSSNODE logical name found?
      OE A6 7C 068E 2294      BNEQ 70$      ; Br if yes
      16 A6 7C 0690 2295      CLRQ MNR_SYIST_NODENAME(R6) ; Otherwise, clear out node name
      27 11 0693 2296      CLRQ MNR_SYIST_NODENAME+8(R6) ; .....
      0696 2297      BRB CNI_SUCC      ; ... and go exit
      0698 2298
      0698 2299      ;
      0698 2300      ; A node name has been found. Now eliminate leading underscore and
      0698 2301      ; trailing double colons if present.
      0698 2302      ;
      0698 2303      ; At this point, R3 = addr of first character of node name, and
      0698 2304      ; R4 = addr of a non-zero word length field
      0698 2305      ;
      0698 2306
      0698 2307 70$:
      0698 2308
      63 5F 8F 91 0698 2309      CMPB #^A/_/, (R3)      ; Is 1st character of string an ??
      04 12 069C 2310      BNEQU 80$      ; No -- go check for trailing colons
      53 D6 069E 2311      INCL R3      ; Yes -- redefine string address ...
      64 B7 06A0 2312      DECW (R4)      ; ... and length to eliminate _
      55 55 64 3C 06A2 2313 80$:
      FE A3 45 9E 06A2 2314      MOVZWL (R4),R5      ; Get current length of string
      65 3A 3A 8F B1 06A5 2315      MOVAB -2(R3),R5      ; Compute addr of end of string - 2
      03 12 06AF 2316      CMPW #^A/::/, (R5)      ; Are last 2 characters colons?
      64 02 A2 06B1 2317      BNEQU 90$      ; No -- go move string to SYI
      OE A6 64 90 06B4 2318      SUBW2 #2,(R4)      ; Yes -- shorten string by 2 bytes
      OF A6 OF 00 63 64 2C 06B4 2319 90$:
      06B8 2320      MOVAB (R4),MNR_SYIST_NODENAME(R6); Get length of node name
      06BF 2321      MOVAB (R4), (R3), #0, #T5, -
      06BF 2322      MNR_SYIST_NODENAME+1(R6) ; ... and actual name string
      06BF 2323
      50 00000000'EF D0 06BF 2324 CNI_SUCC:
      06BF 2325      MOVL NORMAL,R0      ; Indicate success
      06C6 2326 CNI_RET:
      06C6 2327      RET      ; Return
```



```
06C7 2329 .SBTTL ADV_HOM_ITEM - Advance to next display item for homog
06C7 2330
06C7 2331 :++
06C7 2332 :
06C7 2333 : FUNCTIONAL DESCRIPTION:
06C7 2334 :
06C7 2335 : This routine is called by PL/I routines to advance the
06C7 2336 : current (homogeneous) class to the next item for display.
06C7 2337 :
06C7 2338 : CALLING SEQUENCE:
06C7 2339 :
06C7 2340 : CALLS #1,ADV_HOM_ITEM
06C7 2341 :
06C7 2342 : INPUTS:
06C7 2343 :
06C7 2344 : 4(AP) - address of CDB pointer for current display class
06C7 2345 :
06C7 2346 : IMPLICIT INPUTS:
06C7 2347 :
06C7 2348 : None
06C7 2349 :
06C7 2350 : OUTPUTS:
06C7 2351 :
06C7 2352 : None
06C7 2353 :
06C7 2354 : IMPLICIT OUTPUTS:
06C7 2355 :
06C7 2356 : CDX$B_IDISCONSEC and CDX$B_IDISINDEX fields updated to
06C7 2357 : indicate the next item to be displayed.
06C7 2358 :
06C7 2359 : ROUTINE VALUE:
06C7 2360 :
06C7 2361 : SSS_NORMAL
06C7 2362 :
06C7 2363 : SIDE EFFECTS:
06C7 2364 :
06C7 2365 : None
06C7 2366 :
06C7 2367 :--
06C7 2368 :
001C 06C7 2369
06C7 2370 .ENTRY ADV_HOM_ITEM, ^M<R2,R3,R4>
06C9 2371 :
06C9 2372 :
06C9 2373 : Bump CDX$B_IDISCONSEC so that next requested item is chosen for display.
06C9 2374 :
06C9 2375 :
06C9 2376 : MOVL @4(AP),R2 ; Get CDB address
06CD 2377 : MOVL CDB$A_CDX(R2),R2 ; Get CDX address
06D1 2378 : INCB CDX$B_IDISCONSEC(R2) ; Point to next consecutive item
06D4 2379 : CMPB CDX$B_IDISCONSEC(R2), - ; Past final item?
06D9 2380 : CDX$B_IDISCT(R2)
06D9 2381 : BLEQ 10$ ; Br if not
06DB 2382 : MOVB #1,CDX$B_IDISCONSEC(R2) ; Restart consec no. at 1st item
06DF 2383 :
06DF 2384 :
06DF 2385 : Now update CDX$B_IDISINDEX to be in sync with the new value
```

52	04	BC	D0	06C9	2376				
52	32	A2	D0	06CD	2377				
	07	A2	96	06D1	2378				
06	A2	07	A2	91	06D4	2379			
				06D9	2380				
		04	15	06D9	2381				
07	A2	01	90	06DB	2382				
				06DF	2383				
				06DF	2384				
				06DF	2385				


```
06DF 2386 ; of CDX$B_IDISCONSEC.
06DF 2387 ;
06DF 2388 ;
06DF 2389 10$:
51 10 D0 06DF 2390      MOVL    #CDX$S_IBITS,R1      ; Init bit field size
50 D4 06E2 2391      CLRL    R0                    ; Init start position
54 07 A2 9A 06E4 2392
06E4 2393      MOVZBL  CDX$B_IDISCONSEC(R2),R4 ; Get item display consec no.
53 62 51 50 EA 06E8 2394 40$:
06E8 2395      FFS      R0,R1,CDX$W_IBITS(R2),R3 ; Search for next item number
06ED 2396      ; R3 contains item number if found
50 53 01 C1 06ED 2397      ADDL3  #1,R3,R0          ; Compute next starting ...
51 10 50 C3 06F1 2398      SUBL3  R0,#CDX$S_IBITS,R1 ; ... position and field size
FO 54 F5 06F5 2399      SOBGTR  R4,40$           ; Loop until item number is found
06F8 2400
08 A2 53 90 06F8 2401      MOVB    R3,CDX$B_IDISINDEX(R2) ; ... and store it away
06FC 2402
50 01 D0 06FC 2403      MOVL    #SS$_NORMAL,R0      ; Normal status
04 06FF 2404      RET                      ; Return
```



```
0700 2406 .SBTTL COLLECTION - Collect into CURRENT Buffer
0700 2407
0700 2408 :++
0700 2409 :
0700 2410 : FUNCTIONAL DESCRIPTION:
0700 2411 :
0700 2412 : This routine is called to collect data for the current
0700 2413 : class into the buffer pointed to by the CURRENT register
0700 2414 : (R8). The data is obtained either from the running system
0700 2415 : (live collection) or from an input recording file (playback).
0700 2416 :
0700 2417 : CALLING SEQUENCE:
0700 2418 :
0700 2419 : BSBW COLLECTION
0700 2420 :
0700 2421 : INPUTS:
0700 2422 :
0700 2423 : None
0700 2424 :
0700 2425 : IMPLICIT INPUTS:
0700 2426 :
0700 2427 : 4(AP) - address of byte containing class number
0700 2428 :
0700 2429 : Registers:
0700 2430 :
0700 2431 : R6 = CDB pointer
0700 2432 : R7 = MRB pointer
0700 2433 : R8 = CURRENT collection buffer pointer
0700 2434 : R11 = MCA pointer
0700 2435 :
0700 2436 : If non-STD class, R8 has to have been set up before
0700 2437 : entry to this routine.
0700 2438 :
0700 2439 : OUTPUTS:
0700 2440 :
0700 2441 : None
0700 2442 :
0700 2443 : IMPLICIT OUTPUTS:
0700 2444 :
0700 2445 : CURRENT buffer filled with data for this class.
0700 2446 :
0700 2447 : On first collection event, MRBSQ_BEGINNING is loaded with
0700 2448 : the system time of the first collection.
0700 2449 :
0700 2450 : ROUTINE VALUE:
0700 2451 :
0700 2452 : R0 = NORMAL, or error status, if any
0700 2453 :
0700 2454 : SIDE EFFECTS:
0700 2455 :
0700 2456 : Registers R0,R1,R2,R3,R4,R5 destroyed.
0700 2457 :
0700 2458 :--
```



```
0700 2460 COLLECTION:
0700 2461
37 43 A7 03 E1 0700 2462 BBC #MRBSV_PLAYBACK,MRBSW_FLAGS(R7),30$ ; If not collecting from
0705 2463 ; ... a file, go do live collection
0705 2464
08 4B A6 04 E0 0705 2465 BBS #CDB$V_STD,CDB$L_FLAGS(R6),10$ ; Branch if standard class
070A 2466 ;
070A 2467 ; PLAYBACK -- Non-standard Class (PROCESSES)
070A 2468 ;
0080 30 070A 2469 BSBW COLL_NONSTD ; Fill coll buff from playback file
7C 50 E9 070D 2470 BLBC R0, COLL_RSB ; Premature EOF error possible here
52 11 0710 2471 BRB COLL_COMM ; Go join common collection code
0712 2472
0712 2473 ;
0712 2474 ; PLAYBACK -- Standard Class
0712 2475 ;
0712 2476 ; Pick up (with MOVC5) the collection buffer previously placed in a
0712 2477 ; fixed location by the PL/I read routine.
0712 2478 ;
0712 2479
0712 2480 10$:
59 0600 8F BB 0712 2481 PUSHR #^M<R9,R10> ; Save regs 9 and 10
04 AB D0 0716 2482 MOVL MCAS$_INPUT_PTR(R11),R9 ; Get base of read buffer
071A 2483
071A 2484 ;
071A 2485 ; Compute size of collection buffer
071A 2486 ;
071A 2487
5A 20 A6 3C 071A 2488 MOVZWL CDB$W_BLKLEN(R6),R10 ; Get length of a data block
0A 4B A6 05 E1 071E 2489 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),20$ ; Br if hetero
5A 000000C8 8F C4 0723 2490 MULL2 #MAXELTS,R10 ; One data block for each element
SA 08 C0 072A 2491 ADDL2 #MNR_HOM$K_PSIZE,R10 ; Chip in with prefix size
072D 2492 20$:
SA 0D C0 072D 2493 ADDL2 #MNR_CLS$K_HSIZE,R10 ; Add in header size
0730 2494
68 SA 00 69 6B 2C 0730 2495 MOVC5 MCAS$_INPUT_LEN(R11),(R9),#0,R10,(R8) ; Move to CURRENT buffer
0600 8F BA 0736 2496 POPR #^M<R9,R10> ; Restore regs 9 and 10
28 11 073A 2497 BRB COLL_COMM ; Go join common collection code
073C 2498
073C 2499 ;
073C 2500 ; LIVE COLLECTION
073C 2501 ;
073C 2502 ; Create argument list (on stack) for $CMKRNL call to FETCH.
073C 2503 ;
073C 2504 ;
073C 2505 30$:
7E 58 0D C1 073C 2506 ADDL3 #MNR_CLS$K_HSIZE,R8,-(SP) ; Stack addr of beg of variable
0740 2507 ; ... portion of collection buffer
56 DD 0740 2508 PUSHL R6 ; Stack CDB address
02 DD 0742 2509 PUSHL #2 ; Top off list with arg count
54 5E D0 0744 2510 MOVL SP,R4 ; Remember arg list addr for $CMKRNL
0747 2511 ;
0747 2512 ; Fill collection buffer header
0747 2513 ;
68 04 BC 90 0747 2514 MOVB @4(AP),MNR_CLS$B_TYPE(R8) ; Collect class number into buffer
01 A8 B4 074B 2515 CLRW MNR_CLS$W_FLAGS(R8) ; Zero out flags ...
0B A8 B4 074E 2516 CLRW MNR_CLS$W_RESERVED(R8) ; ... and reserved word
```



```

      0751 2517      $CMKRNLS  FETCH,(R4)      ; Call FETCH to fill in rest of buffer
      075E 2518
      5E 0C  C0 075E 2519      ADDL  #12,SP      ; Restore stack
      28 50  E9 0761 2520      BLBC  R0,COLL_RSB ; NOPRIV or other error possible here
      0764 2521
      0764 2522 COLL_COMM:      ; Common code for playback or live collectio
      50 00000000'EF D0 0764 2523      MOVL  NORMAL,R0 ; Indicate successful status
      0B A8 34 AB B0 076B 2524      MOVW  MCASL_CONSEC_REC(R11),MNR_CLS$W_RESERVED(R8)
      0770 2525      ; Store (low-order word of)
      0770 2526      ; ... consec no in coll buff
      0C AB  D5 0770 2527      TSTL  MCASL_COLLCNT(R11) ; First collection?
      17 12 0773 2528      BNEQ  COLL_RSB ; No -- just return
      30 AB 04 BC 91 0775 2529      CMPB  @4(AP),MCASB_FIRSTC(R11) ; First class?
      10 12 077A 2530      BNEQU COLL_RSB ; No -- just return
      67 03 A8 7D 077C 2531      MOVQ  MNR_CLS$Q_STAMP(R8),MRB$Q_BEGINNING(R7)
      0780 2532      ; Yes -- save start time of collection
      07 43 A7 01 E1 0780 2533      BBC  #MRB$V_RECORD,MRB$W_FLAGS(R7),COLL_RSB ; Exit if not recording
      00000000'EF 00 FB 0785 2534      CALLS #0,WRITE_HEADER ; Write recording file header
      078C 2535
      078C 2536 COLL_RSB:
      05 078C 2537      RSB
                                     ; Return
```



```
078D 2539 :  
078D 2540 : Since the PROCESSES class can have class data which spans several  
078D 2541 : records, loop reading all PROCESSES records for this interval,  
078D 2542 : concatenating the data portions of the records into the collection  
078D 2543 : buffer.  
078D 2544 :  
078D 2545 :  
078D 2546 COLL_NONSTD:  
078D 2547 :  
59 0780 8F BB 078D 2548 PUSHR #^M<R7,R8,R9,R10> ; Save registers  
68 59 04 AB D0 0791 2549 MOVL MCAS$_INPUT_PTR(R11),R9 ; Get base of read buffer  
3F 01 69 6B 28 0795 2550 MOVC3 MCAS$_INPUT_LEN(R11),(R9),(R8) ; 1st class rec to CURRENT buffer  
58 00 E1 0799 2551 BBC #MNR_CLSS$_CONT,MNR_CLSS$_FLAGS(R9),30$ ; Done if only 1 class rec  
5A 59 6B C0 079E 2552 ADDL2 MCAS$_INPUT_LEN(R11),R8 ; Point to end of coll buffer  
15 C1 07A1 2553 ADDL3 #<MNR_CLSS$_HSIZE+MNR_PROSL_PSIZE>,R9,R10  
57 D4 07A5 2554 : Point to PROCESSES data in read buff  
07A5 2555 CLRL R7 ; Ensure high-order word is clear ...  
07A7 2556 : ... for later use  
07A7 2557 :  
07A7 2558 : Read the next PROCESSES class record for this interval  
07A7 2559 :  
07A7 2560 :  
07A7 2561 10$:  
7E 00000000'8F D0 07A7 2562 MOVL #SKIP_TO_CLASS,-(SP) ; Indicate next class rec is desired  
5E DD 07AE 2563 PUSHL SP ; Stack indicator for call  
00000000'EF 01 FB 07B0 2564 CALLS #1,READ_INPUT ; Read next class record  
5E 04 C0 07B7 2565 ADDL2 #4,SP ; Pop stack  
09 32 AB 03 E1 07BA 2566 BBC #MCAS$_EOF,MCAS$_FLAGS(R11),20$ ; Continue if not end-of-file  
50 00000000'8F D0 07BF 2567 MOVL #MNR$_PREMEOF,R0 ; Else indicate error ...  
1C 11 07C6 2568 BRB 40$ ; ... and go return  
50 59 0D C1 07C8 2569 20$:  
57 20 A6 60 A5 07CC 2570 ADDL3 #MNR_CLSS$_HSIZE,R9,R0 ; Point to PROCESSES prefix in read buff  
68 6A 57 28 07D1 2571 MULW3 MNR_PROSL_PCTREC(R0),CDB$_BLKLEN(R6),R7 ; Calc size of next move  
58 57 C0 07D1 2572 MOVC3 R7,(R10),(R8) ; Next class rec to CURRENT buffer  
CA 01 A9 00 E0 07D5 2573 ADDL2 R7,R8 ; Point to end of coll buffer  
50 00000000'EF D0 07D8 2574 BBS #MNR_CLSS$_CONT,MNR_CLSS$_FLAGS(R9),10$ ; If cont, go read next  
0780 8F BA 07DD 2575 30$:  
05 07E4 2576 MOVL NORMAL,R0 ; Indicate normal status  
07E4 2577 40$:  
07E4 2578 POPR #^M<R7,R8,R9,R10> ; Restore registers  
07E8 2579 RSB ; Return  
07E9 2580  
07E9 2581
```

```
07E9 2583      .SBTTL FILL_HETERO_STATS - Fill the STATS Buffer
07E9 2584
07E9 2585      :++
07E9 2586      :
07E9 2587      : FUNCTIONAL DESCRIPTION:
07E9 2588      :
07E9 2589      : This routine is called to fill in the STATS buffer
07E9 2590      : for the current (heterogeneous) class, given the current
07E9 2591      : and previous buffers. For COUNT-type data items, each
07E9 2592      : STATS item is computed as CURRENT minus PREVIOUS; for
07E9 2593      : LEVEL-type data items, each STATS item is merely copied
07E9 2594      : from CURRENT. This process may be viewed as "levelizing"
07E9 2595      : the counts.
07E9 2596
07E9 2597      : CALLING SEQUENCE:
07E9 2598      :
07E9 2599      : BSBW FILL_HETERO_STATS
07E9 2600
07E9 2601      : INPUTS:
07E9 2602      :
07E9 2603      : R6 = address of CDB (Class Descriptor Block)
07E9 2604      :
07E9 2605      : R8 = address of CURRENT buffer
07E9 2606      :
07E9 2607      : R9 = address of PREVIOUS buffer
07E9 2608      :
07E9 2609      : R10 = address of buffer block
07E9 2610
07E9 2611      : IMPLICIT INPUTS:
07E9 2612      :
07E9 2613      : PERFTABLE - table describing each data item, indexed by
07E9 2614      : item number ( * entry size)
07E9 2615
07E9 2616      : OUTPUTS:
07E9 2617      :
07E9 2618      : None
07E9 2619
07E9 2620      : IMPLICIT OUTPUTS:
07E9 2621      :
07E9 2622      : STATS buffer is filled.
07E9 2623
07E9 2624      : ROUTINE VALUE:
07E9 2625      :
07E9 2626      : None
07E9 2627
07E9 2628      : SIDE EFFECTS:
07E9 2629      :
07E9 2630      : Registers R0,R1,R2,R3,R4,R5 destroyed.
07E9 2631
07E9 2632      :--
07E9 2633
07E9 2634      : FILL_HETERO_STATS:
07E9 2635
07E9 2636
07E9 2637      : PUSHR #^M<R11>      : Save reg 11
07E9 2638      : MOVL CDB$$_ECOUNT(R6),R0 : Get number of items to fetch
07E9 2639      : MOVL CDB$$_ITMSTR(R6),R1 : Address of item-number string
```

0800 8F BB 07E9 2637
50 18 A6 D0 07ED 2638
51 1C A6 D0 07F1 2639


```
52 59 0D C1 07F5 2640 ADDL3 #MNR_CLSSK_HSIZE,R9,R2 ; Calc start of items for PREVIOUS
53 58 0D C1 07F9 2641 ADDL3 #MNR_CLSSK_HSIZE,R8,R3 ; Calc start of items for CURRENT
54 08 AA D0 07FD 2642 MOVL MBP$A_STATS(R10),R4 ; Get pointer to STATS
5B D4 0801 2643 CLRL R11 ; Clear loop counter
0803 2644 10$:
55 81 9A 0803 2645 MOVZBL (R1)+,R5 ; Get next item number
55 11 C4 0806 2646 MULL #IDB$K_ILENGTH,R5 ; Compute index into IDB table
55 0000'CF45 9E 0809 2647 MOVAB W*PERFTABLE[R5],R5 ; Address of IDB for this item
080F 2648 CASE IDB$W_ISIZE(R5),<20$,30$,40$>,W ; Select on proper size
081A 2649
081A 2650 20$:
0000'8F 0A A5 B1 081A 2651 CMPW IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
0D 12 0820 2652 BNEQ 25$ ; No -- assume level type
64 83 82 83 0822 2653 SUBB3 (R2)+,(R3)+,(R4) ; Compute byte diff into STATS buff
02 18 0826 2654 BGEQ 23$ ; Br if difference OK
64 94 0828 2655 CLRB (R4) ; Counter has decreased; use 0 value
082A 2656 23$:
84 64 9A 082A 2657 MOVZBL (R4),(R4)+ ; Zero-extend to longword
45 11 082D 2658 BRB 50$
082F 2659 25$:
84 83 9A 082F 2660 MOVZBL (R3)+,(R4)+ ; Move CURRENT byte level to STATS
82 95 0832 2661 TSTB (R2)+ ; Auto-increment PREVIOUS buffer
3E 11 0834 2662 BRB 50$
0836 2663 30$:
0000'8F 0A A5 B1 0836 2664 CMPW IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
0D 12 083C 2665 BNEQ 35$ ; No -- assume level type
64 83 82 A3 083E 2666 SUBW3 (R2)+,(R3)+,(R4) ; Compute word diff into STATS buff
02 18 0842 2667 BGEQ 33$ ; Br if difference OK
64 B4 0844 2668 CLRW (R4) ; Counter has decreased; use 0 value
0846 2669 33$:
84 64 3C 0846 2670 MOVZWL (R4),(R4)+ ; Zero-extend to longword
29 11 0849 2671 BRB 50$
084B 2672 35$:
84 83 3C 084B 2673 MOVZWL (R3)+,(R4)+ ; Move CURRENT word level to STATS
82 B5 084E 2674 TSTW (R2)+ ; Auto-increment PREVIOUS buffer
22 11 0850 2675 BRB 50$
0852 2676 40$:
0000'8F 0A A5 B1 0852 2677 CMPW IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
0B 12 0858 2678 BNEQ 45$ ; No -- assume level type
84 83 82 C3 085A 2679 SUBL3 (R2)+,(R3)+,(R4)+ ; Compute long diff into STATS buff
14 18 085E 2680 BGEQ 50$ ; Br if difference OK
FC A4 D4 0860 2681 CLRL -4(R4) ; Counter has decreased; use 0 value
0F 11 0863 2682 BRB 50$
0865 2683 45$:
10 A5 95 0865 2684 TSTB IDB$B_FLAGS(R5) ; Is this a computed item?
05 13 0868 2685 BEQL 47$ ; branch if not
0010 30 086A 2686 BSBW GET_COMPUTED_ITEMS ; otherwise need to do some special calc.
05 11 086D 2687 BRB 50$ ; continue
086F 2688 47$:
84 83 D0 086F 2689 MOVL (R3)+,(R4)+ ; Move CURRENT longword level to STATS
82 D5 0872 2690 TSTL (R2)+ ; Auto-increment PREVIOUS buffer
0874 2691 50$:
8B 5B 50 F2 0874 2692 AOBLSS R0,R11,10$ ; Loop until done
0800 8F BA 0878 2693 POPR #^M<R11> ; Restore reg 11
05 087C 2694 RSB ; ... and return
087D 2695
087D 2696
```


MONITOR
V04-000

- VAX/VMS Performance Monitor Utility^{J 1} 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00 Page 68
FILL_HETERO_STATS - Fill the STATS Buf 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 (35)
087D 2697


```
087D 2699 :++
087D 2700 : GET_COMPUTED_ITEMS
087D 2701 :
087D 2702 :
087D 2703 : Computed items are items which are not recorded, but are transformations
087D 2704 : of other items. This routine makes the appropriate transformations and
087D 2705 : loads the STATS buffer for computed items.
087D 2706 :
087D 2707 : R2 = address of current longword in PREVIOUS buffer
087D 2708 : R3 = address of current longword in CURRENT buffer
087D 2709 : R4 = address of current longword in STAT buffer
087D 2710 :
087D 2711 :
087D 2712 : Currently, the only type of computed item supported by Monitor is
087D 2713 : the percentage item. Here are the steps involved in adding a new percentage
087D 2714 : item:
087D 2715 : - The item is flagged as a percentage by including
087D 2716 :   FLAGS=IDBSM_PCNT in the BLDIDB for the item. The item
087D 2717 :   Type should be LEVEL.
087D 2718 : - The next two items following the percentage in the class
087D 2719 :   will be used to calculate the percentage value. The formula
087D 2720 :   is
087D 2721 :       Percentage value = (Item1 * 100)/Item2
087D 2722 :
087D 2723 : For example, in the case of XQPCACHE percentages,
087D 2724 : Item1 = Cache hits and item2 = (Cache hits + Cache misses)
087D 2725 : The items used to calculate the percentage must be longwords.
087D 2726 :
087D 2727 : - The percentage and item2 will be displayed; item1 will
087D 2728 :   not be displayed.
087D 2729 :
087D 2730 : Computed items currently may only be included in Standard Heterogeneous
087D 2731 : classes.
087D 2732 :
087D 2733 : To add a new type of computed item, define a new bitmask for IDBSB_FLAGS
087D 2734 : (for-instance IDBSM_RATIO) and add the necessary code to do the new
087D 2735 : computation. GET_COMPUTED_ITEMS will be called by FILL_HETERO_STATS anytime
087D 2736 : the IDBSB_FLAGS field is nonzero, so this routine is one place you will want
087D 2737 : to add new code. Depending on the nature of the item, you may need to add
087D 2738 : code elsewhere. (for-instance, TEMPLATE, to prevent display of items used
087D 2739 : in the calculation, and INTAVE to do special processing to obtain the
087D 2740 : average for the item).
087D 2741 :
087D 2742 : Computed items are included in the item count for a class, but are not
087D 2743 : included in the CDBSW_BLKLEN value.
087D 2744 : --
087D 2745 :
087D 2746 : GET_COMPUTED_ITEMS:
087D 2747 :
087D 2748 : If this is a percent, compute the long dif for the following two items
087D 2749 : and do the calculation to end up with a level in the stats buff.
087D 2750 : Also, increment the stats buff pointer but not current or prev.
087D 2751 :
087D 2752 :
087D 2753 :
087D 2754 :
087D 2755 :
```

1D 10 00C0 8F BB 087D 2752 PUSHR #^M<R6,R7> ;save some registers so we can use them
A5 00 E1 0881 2753 BBC #IDBSV_PCNT, IDBSB_FLAGS(R5), 20\$;branch if not a percent
64 D4 0886 2754 CLRL (R4) ; zero longword for this pcnt item in STAT buff
56 63 62 C3 0888 2755 SUBL3 (R2),(R3),R6 ; Current minus previous for item1 into R6


```
57 04 A3 04 A2 C3 088C 2756      SUBL3 4(R2),4(R3),R7 ; Current minus previous for item2 into R7
      57 D5 0892 2757      TSTL R7 ; Is the sum nonzero?
      0B 13 0894 2758      BEQL 10$ ; branch if it is zero to skip percent calc.
      0896 2759 ;
      0896 2760 ; now create a percentage value: (item1 * 100)/(item1 + item2)
      0896 2761 ;
56 00000064 8F C4 0896 2762      MULL2 #100,R6 ;
      64 56 57 C7 089D 2763      DIVL3 R7,R6,(R4) ;move the percentage result into the STAT buff.
      84 D5 08A1 2764 10$:      TSTL (R4)+ ;auto-increment the STAT buffer pointer.
      00C0 8F BA 08A3 2766 20$:      POPR #^M<R6,R7> ;restore the registers
      05 08A7 2768      RSB ;return
      08A8 2769
```



```
.SBTTL FILL_PCSTATS_BUFF - Fill PCSTATS Buffer from STATS Buffer

08A8 2771
08A8 2772
08A8 2773 :++
08A8 2774 :
08A8 2775 : FUNCTIONAL DESCRIPTION:
08A8 2776 :
08A8 2777 : Fill the PCSTATS buffer with integer values representing
08A8 2778 : tenths of percent for each item in the STATS buffer.
08A8 2779 :
08A8 2780 : INPUTS:
08A8 2781 :
08A8 2782 : R6 - CDB pointer
08A8 2783 : R10 - Buffer block pointer
08A8 2784 :
08A8 2785 : IMPLICIT INPUTS:
08A8 2786 :
08A8 2787 : STATS buffer containing levels to be "percentized."
08A8 2788 :
08A8 2789 : OUTPUTS:
08A8 2790 :
08A8 2791 : None
08A8 2792 :
08A8 2793 : IMPLICIT OUTPUTS:
08A8 2794 :
08A8 2795 : PCSTATS buffer filled with integer values representing
08A8 2796 : tenths of percent.
08A8 2797 :
08A8 2798 : ROUTINE VALUE:
08A8 2799 :
08A8 2800 : None
08A8 2801 :
08A8 2802 : SIDE EFFECTS:
08A8 2803 :
08A8 2804 : Registers R0, R1, R3, R4 destroyed.
08A8 2805 :--
08A8 2806 :
08A8 2807 FILL_PCSTATS_BUFF:
08A8 2808
53 18 AA D0 08A8 2809 MOVL MBP$A_PCSTATS(R10),R3 : Load PCSTATS buffer pointer
54 08 AA D0 08AC 2810 MOVL MBP$A_STATS(R10),R4 : Load STATS buffer pointer
50 18 A6 D0 08B0 2811 MOVL CDB$E_COUNT(R6),R0 : Get number of elements
51 51 D4 08B4 2812 CLRL R1 : Clear accumulator
51 84 C0 08B6 2813 10$: ADDL2 (R4)+,R1 : Add next item
51 FA 50 F5 08B9 2814 SOBGTR R0,10$ : Continue until STATS summed
54 08 AA D0 08BC 2815 MOVL MBP$A_STATS(R10),R4 : Re-load STATS buffer pointer
50 18 A6 D0 08C0 2816 MOVL CDB$E_COUNT(R6),R0 : ... and number of elements
51 51 D5 08C4 2817 TSTL R1 : Zero sum?
51 08 12 08C6 2818 BNEQ 30$ : No -- go calc percentages
51 84 D0 08C8 2819 20$: MOVL (R4)+,(R3)+ : Yes -- simply move zeroes
51 FA 50 F5 08CB 2820 SOBGTR R0,20$ : ... into PCSTATS buffer
51 0E 11 08CE 2821 BRB 40$ : ... and go exit
63 84 000003E8 8F C5 08D0 2822 30$: MULL3 #1000,(R4)+,(R3) : Multiply value by 1000 ... and
83 51 C6 08D8 2823 DIVL2 R1,(R3)+ : Divide by sum, leaving tenths of %
F2 50 F5 08DB 2824 SOBGTR R0,30$ : Continue for all items
```

MONITOR
V04-000

- VAX/VMS Performance Monitor Utility^{N 1}
FILL_PCSTATS_BUFF - Fill PCSTATS Buffer

16-SEP-1984 01:59:24
5-SEP-1984 02:01:24

VAX/VMS Macro V04-00
[MONITOR.SRC]MONITOR.MAR;1

Page 72
(37)

05 08DE 2828 40\$:
08DE 2829

RSB

; Return


```
08DF 2831 .SBTTL COMPUTE_STATS - Statistical Computations on STATS
08DF 2832
08DF 2833 :++
08DF 2834 :
08DF 2835 : FUNCTIONAL DESCRIPTION:
08DF 2836 :
08DF 2837 : Replace each count item in the STATS buffer with a computed
08DF 2838 : (floating-point) rate per second. Also, replace each item in
08DF 2839 : the MIN and MAX buffers with the corresponding item from the
08DF 2840 : STATS buffer if it establishes a new minimum or maximum value.
08DF 2841 :
08DF 2842 : INPUTS:
08DF 2843 :
08DF 2844 : R6 - CDB pointer
08DF 2845 : R7 - MRB pointer
08DF 2846 : R8 - CURRENT buffer pointer
08DF 2847 : R9 - PREVIOUS buffer pointer
08DF 2848 : R10 - Buffer block pointer
08DF 2849 : R11 - MCA pointer
08DF 2850 :
08DF 2851 : IMPLICIT INPUTS:
08DF 2852 :
08DF 2853 : STATS buffer
08DF 2854 :
08DF 2855 : HOMOG_TYPE - word containing item type code of current
08DF 2856 : item for homogeneous class.
08DF 2857 :
08DF 2858 : PERFTABLE - table describing each data item, indexed by
08DF 2859 : item number ( * entry size).
08DF 2860 :
08DF 2861 : MCA$ _INTTICKS - clock ticks during interval just finished
08DF 2862 :
08DF 2863 : OUTPUTS:
08DF 2864 :
08DF 2865 : None
08DF 2866 :
08DF 2867 : IMPLICIT OUTPUTS:
08DF 2868 :
08DF 2869 : Each count element in STATS buffer converted to floating rate/second.
08DF 2870 :
08DF 2871 : Elements in MIN and MAX buffers updated if new min and max
08DF 2872 : values were achieved in the interval just completed.
08DF 2873 :
08DF 2874 : ROUTINE VALUE:
08DF 2875 :
08DF 2876 : None
08DF 2877 :
08DF 2878 : SIDE EFFECTS:
08DF 2879 :
08DF 2880 : Registers R0,R1,R2,R3,R4,R5 destroyed.
08DF 2881 : --
08DF 2882 :
08DF 2883 : COMPUTE_STATS:
08DF 2884 :
08DF 2885 : PUSHR #*M<R8,R9> ; Save regs
08E3 2886 :
08E3 2887 : Load registers for upcoming buffer manipulations
```

0300 8F BB


```
08E3 2888 :
08E3 2889 :
08E3 2890 10$:
52 08 AA D0 08E3 2891      MOVL MBP$A_STATS(R10),R2      ; Load addr of first STATS item
58 0C AA D0 08E7 2892      MOVL MBP$A_MIN(R10),R8      ; Load addr of first MIN item
59 10 AA D0 08EB 2893      MOVL MBP$A_MAX(R10),R9      ; Load addr of first MAX item
50 1C A6 D0 08EF 2894      MOVL CDB$A_ITMSTR(R6),R0     ; Load addr of item-number string
53 18 A6 D0 08F3 2895      MOVL CDB$L_ECOUNTR(R6),R3     ; Load number of items in STATS
51 D4 08F7 2896      CLRL R1                          ; Clear loop counter
08F9 2897 20$:
0B 4B A6 05 E1 08F9 2898      BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),25$ ; Br if heterogeneous
0000'8F 00E2'CF B1 08FE 2899      CMPW W^HOMOG_TYPE,#COUNT_TYPE ; Is this homog item a count?
1B 12 0905 2900      BNEQU 30$ ; No -- assume level
33 11 0907 2901      BRB 50$ ; Yes -- go process count
0909 2902 25$:
14 4B A6 00 E1 0909 2903      BBC #CDB$V_CTPRES,CDB$L_FLAGS(R6),30$ ; Skip type check if no counts
55 80 9A 090E 2904      MOVZBL (R0)+,R5 ; Get next item number
55 11 C4 0911 2905      MULL #IDB$K_ILENGTH,R5 ; Compute index into IDB table
55 0000'CF45 9E 0914 2906      MOVAB W^PERFTABLE[R5],R5 ; Address of IDB for this item
0000'8F 0A A5 B1 091A 2907      CMPW IDB$W_TYPE(R5),#COUNT_TYPE ; Is this item a count?
1A 13 0920 2908      BEQLU 50$ ; Yes -- go compute rate
0922 2909 :
0922 2910 : Update MIN and MAX buffers for this item (level).
0922 2911 :
0922 2912 :
0922 2913 30$:
6841 6241 D1 0922 2914      CMPL (R2)[R1],(R8)[R1] ; Check minimum
05 18 0927 2915      BGEQ 40$ ; Branch if not less
6841 6241 D0 0929 2916      MOVL (R2)[R1],(R8)[R1] ; Else insert new minimum
092E 2917 40$:
6941 6241 D1 092E 2918      CMPL (R2)[R1],(R9)[R1] ; Check maximum
33 15 0933 2919      BLEQ 70$ ; Branch if not more
6941 6241 D0 0935 2920      MOVL (R2)[R1],(R9)[R1] ; Else insert new maximum
2C 11 093A 2921      BRB 70$ ; ... and go loop
093C 2922 :
093C 2923 : Compute rate/second for this count item, replacing count in
093C 2924 : STATS buffer.
093C 2925 :
093C 2926 :
093C 2927 50$:
54 6241 4E 093C 2928      CVTLF (R2)[R1],R4 ; Get floating value over interval
55 08 AB 4E 0940 2929      CVTLF MCASL_INTTICKS(R11),R5 ; Get floating ticks over interval
55 000043C8 8F 46 0944 2930      DIVF2 #100,R5 ; Get floating seconds over interval
6241 54 55 47 094B 2931      DIVF3 R5,R4,(R2)[R1] ; Floating rate/second into STATS
0950 2932 :
0950 2933 : Update MIN and MAX buffers for this item (count).
0950 2934 :
0950 2935 :
6841 6241 51 0950 2936      CMPF (R2)[R1],(R8)[R1] ; Check minimum
05 18 0955 2937      BGEQ 60$ ; Branch if not less
6841 6241 50 0957 2938      MOVF (R2)[R1],(R8)[R1] ; Else insert new minimum
095C 2939 60$:
6941 6241 51 095C 2940      CMPF (R2)[R1],(R9)[R1] ; Check maximum
05 15 0961 2941      BLEQ 70$ ; Branch if not more
6941 6241 50 0963 2942      MOVF (R2)[R1],(R9)[R1] ; Else insert new maximum
0968 2943 70$:
8D 51 53 F2 0968 2944      AOBLSS R3,R1,20$ ; Loop for each item in STATS
```


MONITOR
V04-000

D 2
- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
COMPUTE_STATS - Statistical Computations 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 75
(38)

0300 8F BA 096C 2945 POPR #^M<R8,R9> ; Restore regs
05 096C 2946 RSB ; Return
0970 2947

```
0971 2949 .SBTTL UPD_PC_MIN_MAX - Update Percent Min/Max Buffers
0971 2950
0971 2951 :++
0971 2952 :
0971 2953 : FUNCTIONAL DESCRIPTION:
0971 2954 :
0971 2955 : Replace each item in the PCMIN and PCMAX buffers with the
0971 2956 : corresponding item from the PCSTATS buffer if it establishes
0971 2957 : a new minimum or maximum value.
0971 2958 :
0971 2959 : CALLING SEQUENCE:
0971 2960 :
0971 2961 : BSBW UPD_PC_MIN_MAX
0971 2962 :
0971 2963 : INPUTS:
0971 2964 :
0971 2965 : R6 - CDB pointer
0971 2966 : R10 - Buffer block pointer
0971 2967 : R11 - MCA pointer
0971 2968 :
0971 2969 : IMPLICIT INPUTS:
0971 2970 :
0971 2971 : PCSTATS buffer containing percent values derived from
0971 2972 : most recently collected levels.
0971 2973 :
0971 2974 : OUTPUTS:
0971 2975 :
0971 2976 : None
0971 2977 :
0971 2978 : IMPLICIT OUTPUTS:
0971 2979 :
0971 2980 : Items in PCMIN and PCMAX buffers updated accordingly.
0971 2981 :
0971 2982 : ROUTINE VALUE:
0971 2983 :
0971 2984 : None
0971 2985 :
0971 2986 : SIDE EFFECTS:
0971 2987 :
0971 2988 : Registers R0,R1,R2,R3,R4,R5 destroyed.
0971 2989 :--
0971 2990 :
0971 2991 :
0971 2992 UPD_PC_MIN_MAX:
0971 2993 :
53 18 AA D0 0971 2994 MOVL MBP$A_PCSTATS(R10),R3 ; Load PCSTATS ptr
54 1C AA D0 0975 2995 MOVL MBP$A_PCMIN(R10),R4 ; Load PCMIN ptr
55 20 AA D0 0979 2996 MOVL MBP$A_PCMAX(R10),R5 ; Load PCMAX ptr
50 18 A6 D0 097D 2997 MOVL CDB$E_COUNT(R6),R0 ; Get element count
52 D4 0981 2998 CLRL R2 ; Clear loop counter
```



```

0983 3000 :
0983 3001 : Replace minimum and maximum (if necessary) for each item
0983 3002 :
0983 3003 :
0983 3004 10$:
6442 6342 D1 0983 3005 CMPL (R3)[R2],(R4)[R2] ; Check minimum
        05 18 0988 3006 BGEQ 20$ ; Branch if not less
6442 6342 D0 098A 3007 MOVL (R3)[R2],(R4)[R2] ; Else insert new minimum
        05 15 098F 3008 20$:
6542 6342 D1 098F 3009 CMPL (R3)[R2],(R5)[R2] ; Check maximum
        05 15 0994 3010 BLEQ 30$ ; Branch if not more
6542 6342 D0 0996 3011 MOVL (R3)[R2],(R5)[R2] ; Else insert new maximum
        05 15 099B 3012 30$:
E4 52 50 F2 099B 3013 AOBLSS R0,R2,10$ ; Loop for each item in PCSTATS
        05 099F 3014
        05 099F 3015 RSB ; Return
  
```

```

09A0 3017      .SBTTL DISPLAY_INIT - Init for Display Output
09A0 3018
09A0 3019      :++
09A0 3020      :
09A0 3021      : FUNCTIONAL DESCRIPTION:
09A0 3022      :
09A0 3023      :     This routine performs initialization for DISPLAY output.
09A0 3024      :
09A0 3025      : CALLING SEQUENCE:
09A0 3026      :
09A0 3027      :     CALLS #0, DISPLAY_INIT
09A0 3028      :
09A0 3029      : INPUTS:
09A0 3030      :
09A0 3031      :     None.
09A0 3032      :
09A0 3033      : IMPLICIT INPUTS:
09A0 3034      :
09A0 3035      :     MRBPTR - pointer to MRB (Monitor Request Block)
09A0 3036      :     MCAPTR - pointer to MCA (Monitor Communication Area)
09A0 3037      :
09A0 3038      : OUTPUTS:
09A0 3039      :
09A0 3040      :     BARCHAR loaded with appropriate bar character.
09A0 3041      :
09A0 3042      :     SYSOUT_TYPE loaded with display output device class.
09A0 3043      :
09A0 3044      :     The four classes are:
09A0 3045      :
09A0 3046      :         DEC CRT
09A0 3047      :         VT5X
09A0 3048      :         OTHER VID
09A0 3049      :         HARDCOPY (including disk file)
09A0 3050      :
09A0 3051      :     MCASV_VIDEO set if display device is a video terminal.
09A0 3052      :
09A0 3053      :     MCASV_GRAPHICS set if display device is a VT55.
09A0 3054      :
09A0 3055      :     ATTRIBMSK (DEC_CRT video attribute mask) cleared.
09A0 3056      :
09A0 3057      : IMPLICIT OUTPUTS:
09A0 3058      :
09A0 3059      :     None.
09A0 3060      :
09A0 3061      : ROUTINE VALUE:
09A0 3062      :
09A0 3063      :     R0 = NORMAL or failing status from SCRPKG routine.
09A0 3064      :
09A0 3065      : SIDE EFFECTS:
09A0 3066      :
09A0 3067      :     None
09A0 3068      :
09A0 3069      : REGISTER USAGE:
09A0 3070      :
09A0 3071      :     R7  = MRB pointer
09A0 3072      :     R11 = MCA pointer
09A0 3073      :
  
```



```
00000000'CF 272B D4 09A0 3074 :--
00000000'EF 5B 00 43 A7 08 09A0 3075
00EC 30 09A0 3076
088C 09A0 3077 .ENTRY DISPLAY_INIT, ^M<R2,R3,R7,R11>
09A2 3078
09A2 3079 C.LRL W^ATTRIBMSK ; Turn off DEC CRT attributes
09A6 3080 MOVL MRBPTR,R7 ; Load MRB pointer
09AD 3081 MOVL MCAPTR,R11 ; Load MCA pointer
09B4 3082 BBSS #MRB$V_DIS_CL_REQ,MRB$W_FLAGS(R7),5$ ;
09B9 3083 5$: BSBW COMMON_INIT ; Indicate display cleanup required
09B9 3084 ; Do initialization common with SUMMARY
09BC 3085
09BC 3086 : Establish SCRPKG output stream
09BC 3087 :
09BC 3088 :
09BC 3089
09BC 3090 TSTL MRB$A_DISPLAY(R7) ; Is there a display file-spec?
09BF 3091 BEQL 10$ ; No -- continue
09C1 3092 PUSHL MRB$A_DISPLAY(R7) ; Yes -- stack it for SET_OUTPUT
09C4 3093 PUSHL #1 ; ... along with a stream identifier
09C6 3094 CALLS #2,G^SCR$SET_OUTPUT ; Establish output stream
09CD 3095 BLBS R0,10$ ; Continue if status OK
09D0 3096 BRW 110$ ; Go exit if SCR$SET_OUTPUT failed
09D3 3097
09D3 3098 : Establish bar character, sysout device type.
09D3 3099 :
09D3 3100 :
09D3 3101 : Set MCA$V_VIDEO bit if a video terminal.
09D3 3102 :
09D3 3103 : Set MVA$V_GRAPHICS if a VT55 terminal.
09D3 3104 :
09D3 3105 :
09D3 3106 10$:
0005'CF 2A 90 09D3 3107 MOVB #DEF_BAR,W^BARCHAR ; Start out with default bar char
2615'CF 03 90 09D8 3108 MOVB #OTHER_VID,W^SYSOUT_TYPE ; ... and assume 'other video' sysout type
09DD 3109 ALLOC 9,R1,R2 ; Allocate 9 bytes for screen info
09EA 3110 PUSHL R2 ; Stack addr of screen info buffer
00000000'GF 52 DD 09EA 3110 CALLS #1,G^SCR$SCREEN_INFO ; Get screen info (dev type)
03 50 01 FB 09EC 3111 BLBS R0,20$ ; Continue if status OK
0061 31 09F3 3112 BRW 110$ ; Go exit if SCR$SCREEN_INFO failed
09F6 3113
09F9 3114 20$: MOVB SCR$B_DEVTYPE(R2),R3 ; Save SYS$OUTPUT device type
09F9 3115 BBC #SCR$V_SCREEN,SCR$L_FLAGS(R2),30$ ; If not video, keep going
07 62 00 E1 09FD 3116 BBSS #MCA$V_VIDEO,MCA$W_FLAGS(R11),40$ ; Otherwise, indicate video
09 32 AB 04 E2 0A01 3117 BRB 40$ ; ... and continue
07 11 0A06 3118
0A08 3119 30$: MOVB #HARDCOPY,W^SYSOUT_TYPE ; Set hardcopy type
2615'CF 02 90 0A08 3120 BRB 100$ ; ... and go take def bar char
3B 11 0A0D 3121
0A0F 3122 40$: CMPB #DT$_VT55,R3 ; Is it a VT55 ?
53 41 8F 91 0A0F 3123 BNEQU 60$ ; No -- go check for other types
00 32 AB 05 E2 0A15 3125 BBSS #MCA$V_GRAPHICS,MCA$W_FLAGS(R11),50$ ; Yes -- indicate VT55-style graphics
0A1A 3126
0A1A 3127 50$: MOVB #VT5X,W^SYSOUT_TYPE ; Indicate VT5x series
2615'CF 01 90 0A1A 3128 BRB 90$ ; Go set special bar char
07 62 06 E1 0A1F 3129 BRB 90$
0A21 3130 60$: BBC #SCR$V_DECCRT,SCR$L_FLAGS(R2),70$ ; If not DEC CRT, keep going
```



```
2615'CF 00 90 0A25 3131      MOVB  #DEC_CRT,W^SYSOUT_TYPE ; Set DEC CRT
          18 11 0A2A 3132      BRB   90$                ; ... and go set special bar char
53 60 8F 91 0A2C 3133 70$:    CMPB  #DT$_VT100,R3        ; Is it a VT100 ?
          07 12 0A30 3134      BNEQU 80$                ; No -- more checking
2615'CF 00 90 0A32 3135      MOVB  #DEC_CRT,W^SYSOUT_TYPE ; Yes -- set DEC CRT
          0B 11 0A37 3136      BRB   90$                ; ... and go set special bar char
          0A39 3137 80$:      CMPB  #DT$_VT52,R3        ; Is it a VT52 ?
53 40 8F 91 0A39 3138      BNEQU 100$                ; No -- take def bar char and type
          0B 12 0A3D 3139      MOVB  #VT5X,W^SYSOUT_TYPE ; Yes -- indicate VT5x series
2615'CF 01 90 0A3F 3140      MOVB  #VID_BAR,W^BARCHAR    ; Indicate special bar char
0005'CF 61 8F 90 0A44 3141 90$:
          0A4A 3142
          0A4A 3143 100$:    BSBW  MOVE_BARS            ; Move bar char into several places
          009A 30 0A4A 3144
          0A4D 3145
          0A4D 3146
          0A4D 3147
          0A4D 3148 : Kick off buffering mode for the Screen Package
          0A4D 3149 :
          0A4D 3150
000020AF'EF 7F 0A4D 3151      PUSHAQ SCRDCS            ; Push this routine's buffer addr
00000000'GF 01 FB 0A53 3152      CALLS #1,G^LIB$SET_BUFFER ; Set buffering mode
          0A5A 3153 110$:
          04 0A5A 3154      RET                        ; Return with R0 = status
```



```
0A5B 3156 .SBTTL SUMMARY_INIT - Init for Summary Output
0A5B 3157
0A5B 3158 :++
0A5B 3159 :
0A5B 3160 : FUNCTIONAL DESCRIPTION:
0A5B 3161 :
0A5B 3162 : This routine performs initialization for SUMMARY output.
0A5B 3163 :
0A5B 3164 : CALLING SEQUENCE:
0A5B 3165 :
0A5B 3166 : CALLS #0, SUMMARY_INIT
0A5B 3167 :
0A5B 3168 : INPUTS:
0A5B 3169 :
0A5B 3170 : None.
0A5B 3171 :
0A5B 3172 : IMPLICIT INPUTS:
0A5B 3173 :
0A5B 3174 : MRBPTR - pointer to MRB (Monitor Request Block)
0A5B 3175 :
0A5B 3176 : OUTPUTS:
0A5B 3177 :
0A5B 3178 : BARCHAR loaded with appropriate bar character.
0A5B 3179 :
0A5B 3180 : SYSOUT_TYPE loaded with display output device type.
0A5B 3181 :
0A5B 3182 : ATTRIBMSK (DEC_CRT video attribute mask) cleared.
0A5B 3183 :
0A5B 3184 : IMPLICIT OUTPUTS:
0A5B 3185 :
0A5B 3186 : None.
0A5B 3187 :
0A5B 3188 : ROUTINE VALUE:
0A5B 3189 :
0A5B 3190 : R0 = NORMAL or failing status from SCRPKG routine.
0A5B 3191 :
0A5B 3192 : SIDE EFFECTS:
0A5B 3193 :
0A5B 3194 : None
0A5B 3195 :
0A5B 3196 : REGISTER USAGE:
0A5B 3197 :
0A5B 3198 : R7 = MRB pointer
0A5B 3199 :
0A5B 3200 :--
0A5B 3201 :
0A5B 3202 :
0080 0A5B 3203 .ENTRY SUMMARY_INIT, ^M<R7>
0A5D 3204
57 272B'CF D4 0A5D 3205 CLRL W^ATTRIBMSK ; Turn off DEC CRT attributes
00000000'EF D0 0A61 3206 MOVL MRBPTR,R7 ; Load MRB pointer
00 43 A7 09 E2 0A68 3207 BBSS #MRB$V_SUM_CL_REQ,MRB$W_FLAGS(R7), $
03 43 A7 00 E0 0A6D 3208 5$: ; Indicate summary cleanup required
0033 30 0A6D 3209 BBS #MRB$V_DISPLAY,MRB$W_FLAGS(R7),10$ ; Skip init if already done
0A72 3210 BSBW COMMON_INIT ; Do initialization common with DISPLAY
0A75 3211
0A75 3212 ;
```

```
0A75 3213 ; Establish SCRPKG output stream
0A75 3214 ;
0A75 3215 ;
0A75 3216 10$:
28 A7 DD 0A75 3217 PUSHL MRB$A_SUMMARY(R7) ; Stack SUMMARY filespec for SET_OUTPUT
01 DD 0A78 3218 PUSHL #1 ; ... along with a stream identifier
00000000'GF 02 FB 0A7A 3219 CALLS #2,G^SCR$SET_OUTPUT ; Establish output stream
03 50 E8 0A81 3220 BLBS R0,20$ ; Continue if status OK
0020 31 0A84 3221 BRW 40$ ; Go exit if SCR$SET_OUTPUT failed
0A87 3222 ;
0A87 3223 ;
0A87 3224 ; Establish bar character, sysout device type.
0A87 3225 ;
0A87 3226 ;
0A87 3227 20$:
0005'CF 2A 90 0A87 3228 MOVB #DEF_BAR,W^BARCHAR ; Use default bar character
59 10 0A8C 3229 BSBB MOVE_BARS ; Move bar char into several places
00002615'EF 02 90 0A8E 3230 MOVB #HARDCOPY,SYSOUT_TYPE ; Treat SYS$OUTPUT dev type as hardcopy
00 43 A7 05 E2 0A95 3231 BBSS #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R7),30$
0A9A 3232 ; Indicate output to file
0A9A 3233 30$:
0A9A 3234 ;
0A9A 3235 ;
0A9A 3236 ; Kick off buffering mode for the Screen Package
0A9A 3237 ;
0A9A 3238 ;
000020AF'EF 7F 0A9A 3239 PUSHAQ SCR$DSC ; Push this routine's buffer addr
00000000'GF 01 FB 0AA0 3240 CALLS #1,G^LIB$SET_BUFFER ; Set buffering mode
0AA7 3241 40$:
04 0AA7 3242 RET ; Return with R0 = status
```



```
008D'CF 01 D0 0AA8 3244 COMMON_INIT:
0AA8 3245
0AA8 3246 :
0AA8 3247 : Do initialization for DISPLAY and SUMMARY options.
0AA8 3248 :
0AA8 3249 :
0AA8 3250 MOVL #SS$_NORMAL,W^PTS_STAT ; Start off request with clean status
0AAD 3251
0AAD 3252 :
0AAD 3253 : Set up footing display line with appropriate words
0AAD 3254 :
0AAD 3255
23F7'CF 23F5'CF DE 0AAD 3256 MOVAL W^BLANK_STR,W^FOOTP ; Indicate blank string for PLAYBACK
07 43 A7 03 E1 0AB4 3257 BBC #MRB$_PLAYBACK,MRB$_FLAGS(R7),10$ ; Continue if no PLAYBACK
23F7'CF 23C4'CF DE 0AB9 3258 MOVAL W^PLAY_STR,W^FOOTP ; Indicate PLAYBACK string in footing
OAC0 3259 10$:
23FB'CF 23F5'CF DE 0AC0 3260 MOVAL W^BLANK_STR,W^FOOTS ; Indicate blank string for SUMMARY
07 43 A7 02 E1 0AC7 3261 BBC #MRB$_SUMMARY,MRB$_FLAGS(R7),20$ ; Continue if no SUMMARY
23FB'CF 23D3'CF DE 0ACC 3262 MOVAL W^SUMM_STR,W^FOOTS ; Indicate SUMMARY string in footing
OAD3 3263 20$:
23FF'CF 23F5'CF DE 0AD3 3264 MOVAL W^BLANK_STR,W^FOOTR ; Indicate blank string for RECORD
07 43 A7 01 E1 0ADA 3265 BBC #MRB$_RECORD,MRB$_FLAGS(R7),30$ ; Continue if no RECORD
23FF'CF 23E5'CF DE 0ADF 3266 MOVAL W^REC_STR,W^FOOTR ; Indicate RECORD string in footing
OAE6 3267 30$:
05 OAE6 3268 RSB ; Return
OAE7 3269
OAE7 3270
OAE7 3271 MOVE_BARS:
OAE7 3272 MOVB W^BARCHAR,W^TOPBAR ; Move bar char into several places
00000000'EF 0005'CF 90 OAE7 3273 MOVB W^BARCHAR,BAR1 ; Move bar char into TOP display line
00000000'EF 0005'CF 90 OAE7 3274 MOVB W^BARCHAR,BAR2 ; ... and into SYSTEM class FAO str
00000000'EF 0005'CF 90 0AF7 3275 MOVB W^BARCHAR,BAR3 ; .....
00000000'EF 0005'CF 90 0B00 3276 MOVB W^BARCHAR,BAR4 ; .....
00000000'EF 0005'CF 90 0B09 3277 MOVB W^BARCHAR,BAR5 ; .....
00000000'EF 0005'CF 90 0B12 3278 MOVB W^BARCHAR,BAR6 ; .....
00000000'EF 0005'CF 90 0B1B 3279 MOVB W^BARCHAR,BAR7 ; .....
00000000'EF 0005'CF 90 0B24 3280 MOVB W^BARCHAR,BAR8 ; .....
00000000'EF 0005'CF 90 0B2D 3281 MOVB W^BARCHAR,BAR9 ; .....
00000000'EF 0005'CF 90 0B36 3282 MOVB W^BARCHAR,BAR10 ; .....
00000000'EF 0005'CF 90 0B3F 3283 MOVB W^BARCHAR,BAR11 ; .....
00000000'EF 0005'CF 90 0B48 3284
00000085'EF 00000000'EF DE 0B51 3285 MOVAL SYS_BOX_STR_G,SYS_BOX_STR_ADDR ; Choose graphic box str (SYSTEM)
00000083'EF 00000000'EF B0 0B5C 3286 MOVW SYS_BOX_STR_LEN_G,SYS_BOX_STR_LEN ; ... and its length
2A 00000005'EF 91 0B67 3287 CMPB BARCHAR,#DEF_BAR ; This terminal use the def bar char?
16 12 0B6E 3288 BNEQ 10$ ; No, all done
00000085'EF 00000000'EF DE 0B70 3289 MOVAL SYS_BOX_STR_H,SYS_BOX_STR_ADDR ; Choose hardcopy box str
00000083'EF 00000000'EF B0 0B7B 3290 MOVW SYS_BOX_STR_LEN_H,SYS_BOX_STR_LEN ; ... and its length
05 0B86 3291 10$:
0B86 3292 RSB
```

```

0B87 3294 .SBTTL FILL_DISP_BUFF - Fill Display Buffer
0B87 3295
0B87 3296 :++
0B87 3297
0B87 3298 : FUNCTIONAL DESCRIPTION:
0B87 3299
0B87 3300 : This routine is called to fill the display buffer with values
0B87 3301 : to be presented to FAOL for display of the screen for the current
0B87 3302 : class. The address of the CDB for the current class is passed
0B87 3303 : as the first parameter to this routine. The second parameter is
0B87 3304 : the address of a quadword into which this routine will store the
0B87 3305 : time stamp from the most recent collection buffer.
0B87 3306
0B87 3307 : CALLING SEQUENCE:
0B87 3308
0B87 3309 : CALLS #2,FILL_DISP_BUFF
0B87 3310
0B87 3311 : INPUTS:
0B87 3312
0B87 3313 : 4(AP) - address of a pointer to the CDB (Class Descriptor Block)
0B87 3314 : for the class to display.
0B87 3315
0B87 3316 : 8(AP) - address of quadword in which to store the time
0B87 3317 : stamp from the most recent collection buffer.
0B87 3318
0B87 3319 : IMPLICIT INPUTS:
0B87 3320
0B87 3321 : MCAPTR - pointer to MCA (Monitor Communication Area)
0B87 3322
0B87 3323 : MRBPTR - pointer to MRB (Monitor Request Block)
0B87 3324
0B87 3325 : PERFTABLE - table describing each data item, indexed by
0B87 3326 : item number ( * entry size)
0B87 3327
0B87 3328 : FAOSTK - buffer into which to store values for later FAOL call.
0B87 3329
0B87 3330 : OUTPUTS:
0B87 3331
0B87 3332 : Quadword pointed to by 8(AP) is filled with time stamp from
0B87 3333 : most recent collection buffer.
0B87 3334
0B87 3335 : IMPLICIT OUTPUTS:
0B87 3336
0B87 3337 : Display buffer (FAOSTK) buffer is filled.
0B87 3338
0B87 3339 : For the non-standard class (PROCESSES),
0B87 3340 : MCA$$_PROC_DISP is filled with the count
0B87 3341 : of processes to display.
0B87 3342
0B87 3343 : ROUTINE VALUE:
0B87 3344
0B87 3345 : RO = $$$_NORMAL
0B87 3346
0B87 3347 : SIDE EFFECTS:
0B87 3348
0B87 3349 : None
0B87 3350 :

```



```
OB87 3351 : REGISTER USAGE:
OB87 3352 :
OB87 3353 : R5 = pointer to current longword in FAOSTK
OB87 3354 : R6 = address of CDB for class to display
OB87 3355 : R7 = scratch
OB87 3356 : R10 = address of buffer block
OB87 3357 : R11 = address of TM4, a temporary stack area
OB87 3358 :
OB87 3359 : Other registers: see below
OB87 3360 :
OB87 3361 : --
OB87 3362 :
OB87 3363 :
OFFC OB87 3364 .ENTRY FILL_DISP_BUFF, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
OB89 3365
56 04 BC D0 OB89 3366 MOVL @4(AP),R6 ; Load CDB pointer
OB8D 3367 ALLOC TM4$K_SIZE,R0,R11 ; Allocate local temp storage
OB9A 3368
5A 2E A6 D0 OB9A 3369 MOVL CDB$A_BUFFERS(R6),R10 ; Load address of buffer block (MBP)
OE 4B A6 05 E1 OB9E 3370 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if not homog class
50 32 A6 D0 OBA3 3371 MOVL CDB$A_CD(X(R6)),R0 ; Get CDX address
50 07 A0 9A OBA7 3372 MOVZBL CDX$B_IDISCONSEC(R0),R0 ; Get no. of curr display item
5A 50 D7 OBAB 3373 DECL R0 ; Decrement to use as index
5A 6A40 D0 OBAD 3374 MOVL (R10)[R0],R10 ; Get MBP ptr for homog class
OB81 3375
OB81 3376 :
OB81 3377 : Return time stamp from most recent collection buffer to caller.
OB81 3378 :
OB81 3379 :
OB81 3380 10$:
04 59 6A D0 OB81 3381 MOVL MBP$A_BUFFERA(R10),R9 ; Assume BufferA is current
4B A6 01 E1 OB84 3382 BBC #CDB$V_SWAPBUF,CDB$L_FLAGS(R6),20$ ; Br if so
59 04 AA D0 OB89 3383 MOVL MBP$A_BUFFERB(R10),R9 ; BufferB is current
OB8D 3384 20$:
08 BC 03 A9 7D OB8D 3385 MOVQ MNR_CLS$Q_STAMP(R9),@8(AP) ; Give current time stamp to caller
6B 18 A6 D0 OBC2 3386
04 AB 1C A6 D0 OBC2 3387 MOVL CDB$L_ECOUNT(R6), - ; Pick up element count
OBC6 3388 TM4$L_ECOUNT(R11)
OBC6 3389 MOVL CDB$A_ITMSTR(R6), - ; ... and item string addr
OBCB 3390 TM4$A_ITMSTR(R11)
OBCB 3391
OBCB 3392 :
OBCB 3393 : Note -- at this point, R10 contains MBP pointer; element count
OBCB 3394 : and item string address are in TM4$L_ECOUNT and TM4$A_ITMSTR.
OBCB 3395 :
OBCB 3396 :
03 4B A6 04 E1 OBCB 3397 BBC #CDB$V_STD,CDB$L_FLAGS(R6),50$ ; Branch if non-standard class
0100 31 OBD0 3398
OBD3 3400 BRW FDB_STD ; Go process standard class
OBD3 3401 50$:
00 42 A6 91 OBD3 3402 CMPB CDB$B_ST(R6),#REG_PROC ; Non-standard class (PROCESSES)
37 13 OBD7 3403 BEQL FDB_REGPROC ; Regular PROCESSES display?
OBD9 3404 ; Yes -- go fill display buffer for it
OBD9 3405 ; No -- TOP PROCESSES display
OBD9 3406 :
OBD9 3407 : Calculate the two quantites BPU and GMIN for use later in computing
```

```

OBD9 3408 ; the size of the bar graph:
OBD9 3409 :
OBD9 3410 :
OBD9 3411 :
OBD9 3412 :
OBD9 3413 :
OE 45 A6 00 E0 OBD9 3414 BBS #CDB$V PERCENT,CDB$W_QFLAGS(R6),60$ ; Check for percent requested
57 3C A6 4E OBD9 3415 CDTLF CDB$L_RANGE(R6),R7 ; No percent -- get floating range for graph
0000000A'EF 38 A6 D0 OBE2 3416 MOVL CDB$L_MIN(R6),GMIN ; Get minimum value for graph
OD 11 OBEA 3417 BRB 70$ ; Join common code
57 00000064 8F 4E OBE2 3418 60$: CDTLF #100,R7 ; 100 is range of percent graph
0000000A'EF D4 OBF3 3419 CLRL GMIN ; 0 is min value of percent graph
OBF9 3420
OBF9 3421 70$: CDTLF #MAXBARS,R8 ; Get max bar chars per line
00000006'EF 58 28 4C OBF9 3422 DIVF3 R7,R8,BPU ; Calculate bar chars per unit of output
58 57 47 OBF9 3423
OC04 3424
000012A7'EF 56 DD OC04 3425 PUSHL R6 ; Stack PROCESSES CDB pointer
01 FB OC06 3426 CALLS #1,FILL_TOP ; Fill display buffer for TOP display
025B 31 OC0D 3427 BRW FDB_RET ; ... and go return
```



```

OC10 3429 :
OC10 3430 : Fill display buffer for non-standard class (PROCESSES)
OC10 3431 : (regular display).
OC10 3432 :
OC10 3433 : Register usage:
OC10 3434 :
OC10 3435 : R0 = scratch
OC10 3436 : R1 = scratch
OC10 3437 : R2 = process index
OC10 3438 : R3 = process count
OC10 3439 : R5 = pointer to current longword in FAOSTK
OC10 3440 : R6 = address of CDB for class to display
OC10 3441 : R7 = scratch
OC10 3442 : R9 = CURRENT collection buffer
OC10 3443 : R10 = address of buffer block
OC10 3444 : R11 = address of TM4, a temporary stack area
OC10 3445 :
OC10 3446 :
OC10 3447 : FDB_REGPROC:
OC10 3448 :
50 53 59 0D C0 OC10 3449 ADDL2 #MNR_CLSS$HSIZE,R9 ; Point to PROCESSES prefix
50 53 04 A9 D0 OC13 3450 MOVL MNR_PROSL_PCTINT(R9),R3 ; Get process count
50 00000000'EF D0 OC17 3451 MOVL MCAPTR,R0 ; Get MCA pointer
50 18 A0 53 D0 OC1E 3452 MOVL R3,MCA$PROC_DISP(R0) ; Save count for the display rtn
50 59 08 C0 OC22 3453 ADDL2 #MNR_PROSL_PSIZE,R9 ; Point to first data block
50 52 01 D0 OC25 3454 MOVL #1,R2 ; Init loop counter
50 55 08 AA D0 OC28 3455 MOVL MBP$A_PR_FAOSTK(R10),R5 ; Init FAO stack (display buffer) pointer
OC2C 3456 :
OC2C 3457 :
OC2C 3458 : Move individual items for this process from current data block
OC2C 3459 : in collection buffer to longwords in FAO stack.
OC2C 3460 :
OC2C 3461 10$:
OC2C 3462 :
50 50 69 D0 OC2C 3463 MOVL MNR_PROSL_IPID(R9),R0 ; Pick up internal PID
20 A6 33 B1 OC2F 3464 CMPW #MNR_PROSL_EPID,CDB$W_BLKLEN(R6) ; See if we have an EPID
50 33 A9 D0 OC33 3465 BGEQ 15$ ; Br if we do not
50 85 50 D0 OC35 3466 MOVL MNR_PROSL_EPID(R9),R0 ; Take the EPID instead
OC39 3467 15$:
OC39 3468 MOVL R0,(R5)+ ; PID to FAO stack
OC3C 3469 :
OC3C 3470 :
OC3C 3471 : Get STATE cstring pointer
OC3C 3472 :
OC3C 3473 :
50 50 08 A9 3C OC3C 3474 MOVZWL MNR_PROSL_STATE(R9),R0 ; Get state number
50 00000000'EF40 D0 OC40 3475 :
50 02 B1 OC40 3476 MOVL STATELIST[R0],R0 ; Get the STATE cstring.
50 08 A9 B1 OC48 3477 CMPW #SCH$C_MWAIT,- ; Is the process in MWAIT?
50 28 12 OC4A 3478 MNR_PROSL_STATE(R9) ;
50 37 B1 OC4C 3479 BNEQ 20$ ; No, keep process' STATE cstring.
50 20 A6 B1 OC4E 3480 CMPW #MNR_PROSL_EFWM,- ; Yes, see if we have an EFWM
50 25 18 OC50 3481 CDB$W_BLKLEN(R6) ;
50 00000000'EF D0 OC52 3482 BGEQ 20$ ; No, simply use MWAIT cstring
50 1F E0 OC54 3483 MOVL MWAITLIST,R0 ; Yes, get the generic MUTEX cstring.
50 19 37 A9 E0 OC5B 3484 BBS #31,- ; Is this is a MUTEX address?
OC5D 3485 MNR_PROSL_EFWM(R9),20$ ; Yes, keep the MUTEX cstring.
```

MONITOR
V04-000

D 3
- VAX/VMS Performance Monitor Utility
FILL_DISP_BUFF - Fill Display Buffer

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 88
(45)

```
50 00000000'EF D0 0C60 3486 MOVL RWAITLIST,R0 ; No, get the generic RWUNK cstring.
      OF B1 0C67 3487 CMPW #RSNS$ MAX,- ; Is this resource wait code defined?
      37 A9 0C69 3488 MNR_PROSL_EFWM(R9) ;
      OC 1B 0C6B 3489 BLEQU 20$ ; No, keep the RWUNK cstring.
      50 37 A9 3C 0C6D 3490 MOVZWL MNR_PROSL_EFWM(R9),R0 ; Yes, get the resource wait number.
50 00000000'EF40 D0 0C71 3491 MOVL RWAITLIST[R0],R0 ; Get the RWccc cstring.
      0C79 3492
      0C79 3493 20$:
      85 50 D0 0C79 3494 MOVL R0,(R5)+ ; ... and move it to FA0 stack
```



```

      0C7C 3496 :
      0C7C 3497 : Get remainder of process items
      0C7C 3498 :
      0C7C 3499 :
      57 0A A9 9A 0C7C 3500 MOVZBL MNR_PROSB_PRI(R9),R7 ; Get priority (31's complement)
      85 1F 57 C3 0C80 3501 SUBL3 R7,#31,(R5)+ ; Complement it and move to FAO stack
      85 0B A9 7D 0C84 3502 MOVQ MNR_PROSO_LNAME(R9),(R5)+ ; Process name cstring to FAO stack
      85 13 A9 7D 0C88 3503 MOVQ MNR_PROSO_LNAME+8(R9),(R5)+ ; .....
      85 F0 A5 9A 0C8C 3504 MOVZBL -16(R5),(R5)+ ; Length of process name to FAO stack
      85 55 13 C3 0C90 3505 SUBL3 #19,R5,(R5)+ ; Address of process name to FAO stack
      85 1B A9 3C 0C94 3506 MOVZWL MNR_PROSW_GPGCNT(R9),(R5)+ ; Global page count to FAO stack
      57 1D A9 1B A9 A1 0C98 3507 ADDW3 MNR_PROSW_GPGCNT(R9),MNR_PROSW_PPGCNT(R9),R7
      0C9E 3508 ; Get sum of global & process page cnts
      85 57 3C 0C9E 3509 MOVZWL R7,(R5)+ ; ... and move to FAO stack
      07 1F A9 00 E0 0CA1 3510 BBS #PCBSV_RES,MNR_PROSL_STS(R9),30$ ; Br if process was resident
      55 10 C0 0CA6 3511 ADDL2 #16,R5 ; Process non-res; skip next 4 longwords
      85 D4 0CA9 3512 CLRL (R5)+ ; Clear CPUTIM ptr to indicate non-res
      16 11 0CAB 3513 BRB 40$ ; ... and continue
      0CAD 3514 30$: ; Resident process
      85 23 A9 D0 0CAD 3515 MOVL MNR_PROSL_DIOCNT(R9),(R5)+ ; DIO count to FAO stack
      85 27 A9 D0 0CB1 3516 MOVL MNR_PROSL_PAGEFLTS(R9),(R5)+ ; Page fault count to FAO stack
      85 00 2B A9 000186A0 8F 7A 0CB5 3517 EMUL #100000,MNR_PROSL_CPUTIM(R9),#0,(R5)+
      85 55 08 C3 0CBF 3518 ; Xlate ticks to quad time val & move to FAO
      0CC3 3519 ; ... and move ptr to it into FAO stack
      50 20 A6 3C 0CC3 3520 40$:
      59 50 C0 0CC7 3521 MOVZWL CDBSW_BLKLEN(R6),R0 ; Get length of a data block
      FF5C 52 01 53 F1 0CCA 3522 ADDL2 R0,R9 ; Point to next (process) data block in coll
      0198 31 0CD0 3523 ACBL R3,#1,R2,10$ ; Loop to move next process to FAO stack
      0CD0 3524 BRW FDB_RET ; All processes done ... go return
```

```
OCD3 3526 :  
OCD3 3527 : Fill display buffer (FAOSTK) for standard classes.  
OCD3 3528 :  
OCD3 3529 : Register usage:  
OCD3 3530 :  
OCD3 3531 : R0 = scratch, address of current item number  
OCD3 3532 : R1 = scratch  
OCD3 3533 : R2 = data item index  
OCD3 3534 : R3 = address of statistics buffer from buffer block  
OCD3 3535 : R4 = address of IDB for current data item  
OCD3 3536 : R5 = pointer to current longword in FAOSTK  
OCD3 3537 : R6 = address of CDB for class to display  
OCD3 3538 : R7 = scratch  
OCD3 3539 : R8 = statistic code (ALL =0, CUR=1, AVE=2, MIN=3, MAX=4)  
OCD3 3540 : R9 = scratch  
OCD3 3541 : R10 = address of buffer block  
OCD3 3542 : R11 = address of TM4, a temporary stack area  
OCD3 3543 :  
OCD3 3544 :  
OCD3 3545 : FDB_STD:  
OCD3 3546 :  
OCD3 3547 :  
OCD3 3548 : For homogeneous class, store number of elements to display  
OCD3 3549 : (for use in later display routines).  
OCD3 3550 :  
OCD3 3551 :  
08 4B A6 05 E1 OCD3 3552 : BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if a heterogeneous class  
OCD8 3553 :  
50 32 A6 D0 OCD8 3554 : MOVL CDB$A_CDX(R6),R0 ; Get CDB extension for HOMOG class  
1C A0 6B D0 OCD8 3555 : MOVL TM4$L_ECOUNTR11), - ; Store number of elts to display  
OCE0 3556 : CDXL_DCOUNTR0)  
OCE0 3557 : 10$:  
6B D5 OCE0 3558 : TSTL TM4$L_ECOUNTR11) ; Any elements to display?  
03 12 OCE2 3559 : BNEQ 20$ ; Br if have some  
0184 31 OCE4 3560 : BRW FDB_RET ; Else go exit if none  
OCE7 3561 : 20$:  
OCE7 3562 :  
OCE7 3563 :  
OCE7 3564 : If AVERAGE or ALL statistic requested, calculate floating elapsed  
OCE7 3565 : seconds from start of monitor request to time of most recent collection  
OCE7 3566 : event. Store it on the stack for later use.  
OCE7 3567 :  
OCE7 3568 :  
00 42 A6 91 OCE7 3569 : CMPB CDB$B_ST(R6),#ALL_STAT ; ALL stats requested ?  
06 13 OCEB 3570 : BEQL 30$ ; Br if yes  
02 42 A6 91 OCED 3571 : CMPB CDB$B_ST(R6),#AVE_STAT ; Was AVERAGE stat requested ?  
29 12 OCF1 3572 : BNEQ 40$ ; No -- skip following calculation  
OCF3 3573 : 30$:  
57 00000000'EF D0 OCF3 3574 : MOVL MRBPTR,R7 ; Get temp pointer to MRB  
51 07 A9 D0 OCFA 3575 : MOVL MNR CLSSQ_STAMP+4(R9),R1 ; Get high order bits of time  
50 03 A9 67 C3 OCFA 3576 : SUBL3 MRB$Q_BEGINNING(R7),MNR CLSSQ_STAMP(R9),R0 ; Compute elapsed time si  
51 04 A7 D9 OD03 3577 : SBWC MRB$Q_BEGINNING+4(R7),R1 ; Get high order difference  
51 50 50 00002710 8F 7B OD07 3578 : EDIV #10000,R0,R0,R1 ; Turn time into milliseconds  
50 50 4E OD10 3579 : CVTLF R0,R0 ; Floating milliseconds  
OC AB 50 0000457A 8F 47 OD13 3580 : DIVF3 #1000,R0,TM4$L_FLTSECS(R11) ; Save floating seconds on stack  
OD1C 3581 : ; ... for later use
```



```

      OD1C 3583 :
      OD1C 3584 : Execute special routines if this is the SYSTEM class
      OD1C 3585 :
      OD1C 3586 :
      OD1C 3587 40$:
39 4B A6 08 E1 OD1C 3588 BBC #CDB$V_SYSCLS,CDB$L_FLAGS(R6),55$ ; Br if not SYSTEM class
   00 42 A6 91 OD21 3589 CMPB CDB$B_ST(R6),#ALL_STAT ; ALL stats requested ?
      18 13 OD25 3590 BEQL 50$ ; Br if yes
      0265 31 OD27 3591 ALLOC <4*ECOUNT_SYS_SINGLE>,R0,R3 ; Single stat -- get dummy buffer
      OD3C 3592 BRW FDB_SYS_SINGLE ; ... and branch to special rtn
      OD3F 3593 50$:
      OD3F 3594 ALLOC <MBP$K_SIZE+<4*4*ECOUNT_SYS_ALL>>,R0,R10
      OD54 3595 ; Alloc dummy MBP and stats buffers
      OD54 3596 ; ... pointed to by R10
      00001209'EF 16 OD54 3597 JSB FDB_SYS_ALL ; Pre-process the SYSTEM class
      OD5A 3598 55$:
      55 00000103'EF DE OD5A 3599
      58 42 A6 9A OD61 3600 MOVAL FAOSTK,R5 ; Load addr of display buffer
      OD65 3601 MOVZBL CDB$B_ST(R6),R8 ; Load requested statistic
      OD65 3602
      OD65 3603 :
      OD65 3604 : If single statistic (not ALL) requested, calculate and store two quantities
      OD65 3605 : for later use:
      OD65 3606 :
      OD65 3607 : BPU - floating longword, no of bar chars per unit of output value
      OD65 3608 : GMIN - integer longword, min value which graph can represent for this class
      OD65 3609 :
      OD65 3610 :
      00 58 D1 OD65 3611 CMPL R8,#ALL_STAT ; ALL requested?
      2B 13 OD68 3612 BEQL 80$ ; Yes -- continue
      OE 45 A6 00 E0 OD6A 3613 BBS #CDB$V_PERCENT,CDB$W_QFLAGS(R6),60$ ; No -- check for percent request
      50 3C A6 4E OD6F 3614 CVTLF CDB$L_RANGE(R6),R0 ; No percent -- get floating range for graph
      0000000A'EF 38 A6 D0 OD73 3615 MOVL CDB$L_MIN(R6),GMIN ; Get minimum value for graph
      OD 11 OD7B 3616 BRB 70$ ; Join common code
      50 00000064 8F 4E OD7D 3617 60$:
      0000000A'EF D4 OD84 3618 CVTLF #100,R0 ; 100 is range of percent graph
      OD8A 3619 CLRL GMIN ; 0 is min value of percent graph
      70$:
      00000006'EF 51 28 4C OD8A 3621 CVTBF #MAXBARS,R1 ; Get max bar chars per line
      51 50 47 OD8D 3622 DIVF3 R0,R1,BPU ; Calculate bar chars per unit of output
      OD95 3623
      OD95 3624 :
      OD95 3625 : For homogeneous class, determine item type for use below.
      OD95 3626 :
      OD95 3627 :
      OD95 3628 80$:
      1C 4B A6 05 E1 OD95 3629 BBC #CDB$V_HOMOG,CDB$L_FLAGS(R6),90$ ; Br if a heterogeneous class
      52 32 A6 D0 OD9A 3630
      OD9A 3631 MOVL CDB$A_CDX(R6),R2 ; Get CDB extension for HOMOG class
      53 08 A2 9A OD9E 3632
      54 04 BB43 9A ODA2 3633 MOVZBL CDX$B_IDISINDEX(R2),R3 ; Get item index for this disp event
      54 54 11 C4 ODA7 3634 MOVZBL @TM4$A_ITMSTR(R11)[R3],R4 ; Load IDB item number
      54 0000'CF44 9E ODA7 3635 MULL2 #IDB$K_ILENGTH,R4 ; Compute index into IDB table
      00E0'CF 0A A4 B0 ODA7 3636 MOVAB W^PERFTABLE[R4],R4 ; Address of IDB for this item
      OD80 3637 MOVW IDB$W_TYPE(R4),W^ITEM_TYPE ; Save item type for use below
```

```

        ODB6 3639 :
        ODB6 3640 : Loop once for each element in this class. Pick up transformed value for
        ODB6 3641 : desired statistic from appropriate buffer within collection buffer
        ODB6 3642 : block. Do computation on transformed value if required, and place
        ODB6 3643 : whole and fractional portions of result into display buffer (FAOSTK).
        ODB6 3644 :
        ODB6 3645 :
        ODB6 3646 90$:
        6B D7 ODB6 3647 DECL TM4$$_ECOUNT(R11) ; Set up number of elements to
        50 04 AB D0 ODB8 3648 ; ... display as a loop limit
        52 D4 ODB8 3649 MOVL TM4$$_ITMSTR(R11),R0 ; Address of item-number string
        ODBC 3650 CLRL R2 ; Clear element index register
        ODBE 3651 FDB_BEG:
        12 4B A6 05 E0 ODBE 3652 BBS #CDB$$_HOMOG,CDB$$_FLAGS(R6),10$ ; Br if a homogeneous class
        54 80 9A ODC3 3653 (R0)+,R4 ; Get next item number
        54 11 C4 ODC6 3654 MULL2 #IDB$$_ILENGTH,R4 ; Compute index into IDB table
        54 0000'CF44 9E ODC9 3655 MOVAB W^PERFTABLE[R4],R4 ; Address of IDB for this item
        00E0'CF 0A A4 E0 ODC9 3656 MOVW IDB$$_TYPE(R4),W^ITEM_TYPE ; Save item type for use below
        ODD5 3657 10$:
        ODD5 3658 CASE R8,<F_ALL,F_CUR,F_AVE,F_MIN,F_MAX>,L ; Select on requested statistic
        ODE3 3659
        ODE3 3660 F_ALL:
        ODE3 3661 F_CUR:
        09 45 A6 00 E0 ODE3 3662 BBS #CDB$$_PERCENT,CDB$$_QFLAGS(R6),10$ ; Br if percent requested
        53 08 AA D0 ODE8 3663 MOVL MBP$$_STATS(R10),R3 ; Load addr of STATS buffer
        0104 30 ODEC 3664 BSBW INTORFL ; Process an integer or floating value
        0A 11 ODEF 3665 BRB 20$ ; ... and continue
        ODF1 3666 10$:
        53 18 AA D0 ODF1 3667 MOVL MBP$$_PCSTATS(R10),R3 ; Load addr of PCSTATS buffer
        59 6342 D0 ODF5 3668 MOVL (R3)[R2],R9 ; Load tenths of % value
        74 10 ODF9 3669 BSB PCTEN ; Process a tenths of % value
        ODFB 3670 20$:
        58 D5 ODFB 3671 TSTL R8 ; Was ALL statistic requested?
        5D 12 ODFD 3672 BNEQ COMMON ; No -- join common code
        ODFD 3673 ; Yes -- continue to AVE
        ODFD 3674 F_AVE:
        09 45 A6 00 E0 ODFD 3675 BBS #CDB$$_PERCENT,CDB$$_QFLAGS(R6),10$ ; Br if percent requested
        53 14 AA D0 OE04 3676 MOVL MBP$$_SUM(R10),R3 ; Load addr of SUM buffer
        007B 30 OE08 3677 BSBW INTAVE ; Process an integer average value
        17 11 OE0B 3678 BRB 20$ ; ... and continue
        OE0D 3679 10$:
        53 24 AA D0 OE0D 3680 MOVL MBP$$_PCSUM(R10),R3 ; Load addr of PCSUM buffer
        51 00000000'EF D0 OE11 3681 MOVL MCAPTR,R1 ; Get MCA pointer
        57 0C A1 01 C3 OE18 3682 SUBL3 #1,MCA$$_COLLCNT(R1),R7 ; Get no of colls, don't count 1st
        59 6342 57 C7 OE1D 3683 DIVL3 R7,(R3)[R2],R9 ; Get average tenths % value
        4B 10 OE22 3684 BSB PCTEN ; ... and process it
        OE24 3685 20$:
        58 D5 OE24 3686 TSTL R8 ; Was ALL statistic requested?
        34 12 OE26 3687 BNEQ COMMON ; No -- join common code
        OE28 3688 ; Yes -- continue to MIN
        OE28 3689 F_MIN:
        09 45 A6 00 E0 OE28 3690 BBS #CDB$$_PERCENT,CDB$$_QFLAGS(R6),10$ ; Br if percent requested
        53 0C AA D0 OE2D 3691 MOVL MBP$$_MIN(R10),R3 ; Load addr of MIN buffer
        00BF 30 OE31 3692 BSBW INTORFL ; Process an integer or floating value
        0A 11 OE34 3693 BRB 20$ ; ... and continue
        OE36 3694 10$:
        53 1C AA D0 OE36 3695 MOVL MBP$$_PCMIN(R10),R3 ; Load addr of PCMIN buffer
```



```

59 6342 D0 0E3A 3696      MOVL      (R3)[R2],R9      ; Load tenths of % value
      2F 10 0E3E 3697      BSB      PCTEN           ; Process a tenths of % value
      58 D5 0E40 3698 20$: TSTL      R8              ; Was ALL statistic requested?
      18 12 0E42 3699      BNEQ     COMMON           ; No -- join common code
      0E44 3701           ; Yes -- continue to MAX
      0E44 3702 F_MAX:
09 45 A6 00 E0 0E44 3703      BBS      #CDB$V_PERCENT,CDB$W_QFLAGS(R6),10$ ; Br if percent requested
53 10 AA D0 0E49 3704      MOVL     MBP$A_MAX(R10),R3 ; Load addr of MAX buffer
      00A3 30 0E4D 3705      BSBW    INTORFL         ; Process an integer or floating value
      0A 11 0E50 3706      BRB      COMMON           ; ... and continue
      0E52 3707 10$:
53 20 AA D0 0E52 3708      MOVL     MBP$A_PCMAX(R10),R3 ; Load addr of PCMAX buffer
59 6342 D0 0E56 3709      MOVL     (R3)[R2],R9      ; Load tenths of % value
      13 10 0E5A 3710      BSB      PCTEN           ; Process a tenths of % value
      0E5C 3711
      0E5C 3712 COMMON:
04 10 A4 00 E1 0E5C 3713      BBC      #IDB$V_PCNT,IDB$B_FLAGS(R4),10$ ;branch if it is not a pcnt
      50 D6 0E61 3714      INCL     R0              ; increment index into item table
      52 D6 0E63 3715      INCL     R2              ; increment index into data buffer
      0E65 3716 10$:
FF53 52 01 6B F1 0E65 3717      ACBL     TM4$L_ECOUNT(R11),#1,R2,FDB_BEG ; Loop once for each element
      0E6B 3718
      0E6B 3719 FDB_RET:
      50 01 D0 0E6B 3720      MOVL     #SS$_NORMAL,R0 ; Indicate success
      04 0E6E 3721      RET              ; ... and return
```

```
0E6F 3723 :  
0E6F 3724 : PCTEN - Transform an integer tenths of percent value to a whole integer  
0E6F 3725 : percent value and an optional integer tenths "remainder"  
0E6F 3726 : value. Place result(s) in display buffer (FAOSTK).  
0E6F 3727 :  
0E6F 3728 : R2 = data item index  
0E6F 3729 : R3 = address of source statistics buffer  
0E6F 3730 : R5 = address of current longword in display buffer  
0E6F 3731 : R7 = scratch  
0E6F 3732 : R8 = code for requested statistic  
0E6F 3733 : R9 = tenths of % value (input)  
0E6F 3734 :  
0E6F 3735 :  
0E6F 3736 : PCTEN:  
85 59 0A C7 0E6F 3737 : DIVL3 #10,R9,(R5)+ ; Compute whole percent value  
58 D5 0E73 3738 : TSTL R8 ; Requested ALL statistics?  
05 13 0E75 3739 : BEQL 10$ ; Yes -- continue  
00B4 30 0E77 3740 : BSBW CALC_BAR ; No -- go do bar graph calcs  
09 11 0E7A 3741 : BRB 20$ ; ... and get out  
57 FC A5 0A C5 0E7C 3742 10$: MULL3 #10,-4(R5),R7 ; Need fraction for tabular display  
85 59 57 C3 0E81 3743 : SUBL3 R7,R9,(R5)+ ; Into display buffer  
0E85 3744 :  
05 0E85 3745 20$: RSB  
0E86 3746 :  
0E86 3747 :  
0E86 3748 :  
0E86 3749 : INTAVE - Transform an integer sum of level values or counts into an  
0E86 3750 : average size/collection or rate/second. Place whole  
0E86 3751 : and optional fractional parts into display buffer (FAOSTK).  
0E86 3752 :  
0E86 3753 : R1 = scratch  
0E86 3754 : R2 = data item index  
0E86 3755 : R3 = address of source statistics buffer  
0E86 3756 : R4 = address of IDB for current item  
0E86 3757 : R5 = address of current longword in display buffer  
0E86 3758 : R7 = scratch  
0E86 3759 : R8 = code for requested statistic  
0E86 3760 : R9 = scratch  
0E86 3761 : R11 = address of TM4  
0E86 3762 : TM4$L_FLTSECS = floating seconds from start of monitor request  
0E86 3763 : to most recent collection event.  
0E86 3764 :  
0E86 3765 :  
25 10 A4 00 E1 0E86 3766 : INTAVE:  
0E86 3767 : BBC #IDB$V_PCNT,IDB$B_FLAGS(R4),7$ ;branch if it is not a percent  
0E8B 3768 :  
0E8B 3769 : ;If it is percent, compute the average based on the sums of item1 and item2  
0E8B 3770 :  
57 00000064 8F 04 A342 D4 0E8B 3771 : CLRL (R5) ;Zero the current longword in Fao buff  
57 57 57 C5 0E8D 3772 : MULL3 4(R3)[R2],#100,R7 ;sum for item1*100 into R7  
59 08 A342 D0 0E97 3773 : CVTLF R7,R7 ;convert to float  
59 59 D5 0E9A 3774 : MOVL 8(R3)[R2],R9 ;sum for (item1+item2) into R9  
09 13 0E9F 3775 : TSTL R9 ;item1 + item2 = 0?  
59 59 4E 0EA1 3776 : BEQL 5$ ;skip divide if so  
57 59 46 0EA3 3777 : CVTLF R9,R9 ;convert to float  
65 57 4A 0EA6 3778 : DIVF R9,R7 ;compute floating avg  
0EA9 3779 : CVTFL R7,(R5) ;stack whole part for fao
```


			85	D5	OEAC	3780	5\$:				
			28	11	OEAC	3781		TSTL	(R5)+		
					OEAE	3782		BRB	25\$		;increment display buff pointer
					OEBO	3783					;and move on
					OEBO	3784	7\$:				
					OEBO	3785		CVTLF	(R3)[R2],R7		; Get floating sum
0000'8F	57	6342		4E	OEBO	3785		CMPW	W^ITEM_TYPE,#COUNT_TYPE		; This item a count?
		00E0'CF		B1	OEBO	3786		BNEQ	10\$		; No -- assume level type
		06		12	OEBO	3787		DIVF	TM4\$L_FLTSECS(R11),R7		; Yes -- get floating avg rate/second
	57	OC AB		46	OEBO	3788		BRB	20\$		; ... and continue
		12		11	OEBO	3789					
					OEBO	3790	10\$:				
					OEBO	3791		MOVL	MCAPTR,R1		; Get MCA pointer
51	00000000'EF			D0	OEBO	3791		SUBL3	#1,MCA\$L_COLLCNT(R1),R9		; Get no of colls, don't count 1st
59	OC A1	01		C3	OEBO	3792		CVTLF	R9,R9		; Get floating no of collections
	59	59		4E	OEBO	3793		DIVF	R9,R7		; Compute floating avg size/collection
	57	59		46	OEBO	3794					
					OEBO	3795	20\$:				
	85	57		4A	OEBO	3796		CVTFL	R7,(R5)+		; Stack whole part for fao
					OEBO	3797	25\$:				
					OEBO	3798		TSTL	R8		; Requested ALL statistics?
					OEBO	3799		BEQL	30\$		; Yes -- continue
					OEBO	3800		BSBW	CALC_BAR		; No -- go do bar graph calcs
					OEBO	3801		BRB	40\$		; ... and get out
					OEBO	3802	30\$:				
	59	FC A5		4E	OEBO	3803		CVTLF	-4(R5),R9		; Get back truncated part
	57	59		42	OEBO	3804		SUBF	R9,R7		; Compute fraction to two ...
57	000043C8	8F		44	OEBO	3805		MULF	#100,R7		; ... digits for tabular display
	85	57		4A	OEBO	3806		CVTFL	R7,(R5)+		; Stack fraction for fao
					OEBO	3807	40\$:				
				05	OEBO	3808		RSB			

```
0EF3 3810
0EF3 3811
0EF3 3812 : INTORFL - Place whole and optional fractional parts of integer value
0EF3 3813 : (level) or floating rate/second value (count) into
0EF3 3814 : display buffer (FAOSTK).
0EF3 3815
0EF3 3816 R2 = data item index
0EF3 3817 R3 = address of source statistics buffer
0EF3 3818 R4 = address of IDB for current item
0EF3 3819 R5 = address of current longword in display buffer
0EF3 3820 R7 = scratch
0EF3 3821 R8 = code for requested statistic
0EF3 3822 R9 = scratch
0EF3 3823
0EF3 3824
0EF3 3825 INTORFL:
0000'8F 00E0'CF B1 0EF3 3826 CMPW W^ITEM_TYPE,#COUNT_TYPE ; Is this item a count?
      11 13 0EFA 3827 BEQL 20$ ; Br if yes
      85 6342 D0 0EFC 3828 MOVL (R3)[R2],(R5)+ ; Move level value to disp buffer
      58 D5 0F00 3829 TSTL R8 ; ALL statistics requested?
      05 13 0F02 3830 BEQL 10$ ; Yes -- continue
      0027 30 0F04 3831 BSBW CALC_BAR ; No -- go do bar graph calcs
      24 11 0F07 3832 BRB 40$ ; ... and get out
      85 D4 0F09 3833 10$: CLRL (R5)+ ; Stack fractional part
      20 11 0F0B 3835 BRB 40$ ; ... and exit
      85 6342 4A 0F0D 3836 20$: CVTFL (R3)[R2],(R5)+ ; Stack whole part of rate (for count)
      58 D5 0F11 3838 TSTL R8 ; ALL statistics requested?
      05 13 0F13 3839 BEQL 30$ ; Yes -- continue
      0016 30 0F15 3840 BSBW CALC_BAR ; No -- go do bar graph calcs
      13 11 0F18 3841 BRB 40$ ; ... and get out
      57 FC A5 4E 0F1A 3842 30$: CVTLF -4(R5),R7 ; Get back rounded part
      59 6342 57 43 0F1E 3844 SUBF3 R7,(R3)[R2],R9 ; Compute fraction
      59 000043C8 8F 44 0F23 3845 MULF #100,R9 ; ... to two digits
      85 59 4A 0F2A 3846 CVTFL R9,(R5)+ ; Stack fractional part
      05 0F2D 3847 40$: RSB ; Return to caller
      05 0F2D 3848
```



```
OF2E 3850 :  
OF2E 3851 : CALC_BAR - Replace integer longword value in display buffer with three  
OF2E 3852 : longwords representing the width of a field to display  
OF2E 3853 : the value (0 or 7), the value, and the number of bar  
OF2E 3854 : characters needed to represent the value in a bar graph.  
OF2E 3855 :  
OF2E 3856 : Note -- if SYSTEM class, merely annex bar count to count  
OF2E 3857 : already in display buffer.  
OF2E 3858 :  
OF2E 3859 : Register Inputs:  
OF2E 3860 :  
OF2E 3861 : R1 = scratch  
OF2E 3862 : R5 = address of current longword in display buffer;  
OF2E 3863 : will be updated to next available longword on output  
OF2E 3864 : R6 = addr of CDB  
OF2E 3865 : R7 = scratch  
OF2E 3866 : R9 = scratch  
OF2E 3867 :  
OF2E 3868 : Implicit Inputs:  
OF2E 3869 :  
OF2E 3870 : GMIN - Integer longword, min value for graph  
OF2E 3871 : BPU - Floating longword, bar chars per unit of value  
OF2E 3872 :  
OF2E 3873 :  
OF2E 3874 :  
OF2E 3875 :  
OF2E 3876 : This subroutine creates 3 longwords in the FAOSTK array for the current  
OF2E 3877 : data item. The current item is represented by a longword integer value  
OF2E 3878 : which will be displayed to the left of the bar in the bar graph for  
OF2E 3879 : the current class. The current item has already been placed in the  
OF2E 3880 : display buffer by the calling routine; R5 has already been advanced to  
OF2E 3881 : the next available longword. This subroutine must replace the value  
OF2E 3882 : longword in the display buffer (FAOSTK) with 3 longwords, leaving R5  
OF2E 3883 : pointing to the next available longword. The three longwords are:  
OF2E 3884 : width of value field (0 or 7), value, and number of bar characters  
OF2E 3885 : needed to represent the value. The number of bargraph characters is  
OF2E 3886 : computed in floating point and then truncated.  
OF2E 3887 :  
OF2E 3888 :  
OF2E 3889 : CALC_BAR:  
35 4B A6 08 E0 OF2E 3890 BBS #CDB$V_SYSCLS,CDB$L_FLAGS(R6),30$ ; Br if SYSTEM class  
57 D4 OF2E 3891 CLRL R7 ; Assume value field width will be 0  
65 FC A5 D0 OF2E 3892 MOVL -4(R5),(R5) ; Move value ahead in display buffer  
03 13 OF2E 3893 BEQL 10$ ; If value zero, go move 0 field width  
57 07 9A OF2E 3894 MOVZBL #7,R7 ; Value is non-zero; field width is 7  
OF2E 3895 10$:  
FC A5 57 D0 OF2E 3896 MOVL R7,-4(R5) ; Move value field width into display buffer  
OF2E 3897 :  
OF2E 3898 : Now calculate number of bars to output  
OF2E 3899 :  
59 85 0000000A'EF 57 D4 OF2E 3900 CLRL R7 ; Assume no bars will be output  
C3 OF2E 3901 SUBL3 GMIN,(R5)+,R9 ; Calc units of value to output and ...  
OF2E 3902 : ... advance R5 to "no of bars" longword  
15 15 OF2E 3903 BLEQ 20$ ; Output no bars if leq zero  
59 59 4E OF2E 3904 CVTFL R9,R9 ; Convert units to floating  
59 00000006'EF 44 OF2E 3905 MULF2 BPU,R9 ; Bars/unit * units => bars to output  
57 59 4A OF2E 3906 CVTFL R9,R7 ; Integer number to output
```

```
57 28 D1 0F5B 3907      CMPL    #MAXBARS,R7      ; Check for upperbound
57 03 18 0F5E 3908      BGEQ    20$              ; Continue if within range
57 28 D0 0F60 3909      MOVL    #MAXBARS,R7      ; Else make it within range
85 57 D0 0F63 3910      MOVL    R7,(R5)+          ; Move number of bars to display buffer ...
3B 11 0F66 3911      BRB      50$              ; ... and advance R5 to next longword
0F66 3912              ; Go return to caller
0F68 3913
0F68 3914
0F68 3915
0F68 3916      ; Special processing for SYSTEM class
0F68 3917
0F68 3918
0F68 3919      30$:
51 00000000'EF DE 0F68 3920      MOVAL    FMT_SYS_SINGLE,R1      ; Get vector of format codes
00'8F 6142 91 0F6F 3921      CMPB     (R1)[R2],#NUMB_ONLY      ; Number only desired?
2D 13 0F74 3922      BEQL     50$              ; Br if yes -- all done
0F76 3923
0F76 3924      ;
0F76 3925      ; NUMB_BAR type ... number and bar desired. Number is already in stack.
0F76 3926      ; Now calculate number of bars to output.
0F76 3927
0F76 3928
57 D4 0F76 3929      CLRL     R7              ; Assume no bars will be output
FC A5 D5 0F78 3930      TSTL     -4(R5)          ; Zero units of value to output?
23 15 0F7B 3931
0F7B 3932      BLEQ     40$              ; Output no bars if leq zero
0F7D 3933
51 00000000'EF DE 0F7D 3934      MOVAL    BU_SYS_SINGLE,R1      ; Get addr of vector of bar ranges
59 59 6142 4E 0F84 3935      CVTLF    (R1)[R2],R9      ; Get floating range for graph
51 1A 4C 0F88 3936      CVTBF    #MAXBARS_SYS,R1      ; Get max bar chars per line
51 59 46 0F8B 3937      DIVF2    R9,R1          ; Calculate bar chars per unit of output
59 FC A5 4E 0F8E 3938      CVTLF    -4(R5),R9      ; Get floating units of value to output
59 51 44 0F92 3939      MULF2    R1,R9          ; Bars/unit * units => bars to output
57 59 4A 0F95 3940      CVTFL    R9,R7          ; Integer number to output
57 1A D1 0F98 3941      CMPL     #MAXBARS_SYS,R7      ; Check for upperbound
03 18 0F9B 3942      BGEQ     40$              ; Continue if within range
57 1A D0 0F9D 3943      MOVL     #MAXBARS_SYS,R7      ; Else make it within range
85 57 D0 0FA0 3944      MOVL     R7,(R5)+          ; Move number of bars to display buffer ...
0FA0 3945      ; ... and advance R5 to next longword
0FA3 3946      40$:
0FA3 3947      MOVL     R7,(R5)+
0FA3 3948      50$:
05 0FA3 3949      RSB
3950      ; Return to caller
```



```
OFA4 3952 :  
OFA4 3953 : FDB_SYS_SINGLE  
OFA4 3954 :  
OFA4 3955 : Fill a dummy statistics buffer from similar buffers of the MODES  
OFA4 3956 : STATES and SYSTEM classes. Then call INTORFL and INTAVE routines  
OFA4 3957 : to transform the data in the buffer to items on the FAOSTK.  
OFA4 3958 :  
OFA4 3959 : This routine is entered with a direct branch, and branches to  
OFA4 3960 : FDB_RET when done to return to original caller.  
OFA4 3961 :  
OFA4 3962 : Inputs:  
OFA4 3963 :  
OFA4 3964 : R0 - R2 scratch  
OFA4 3965 : R3 - address of statistics buffer.  
OFA4 3966 : R4 - R5 scratch  
OFA4 3967 : R6 - address of SYSTEM CDB.  
OFA4 3968 : R7 - R10 scratch  
OFA4 3969 : R11 - address of TM4, a temporary stack area  
OFA4 3970 :  
OFA4 3971 :  
OFA4 3972 : FDB_SYS_SINGLE:  
OFA4 3973 :  
50 00000000'8F DO OFA4 3974 MOVL #<CDB$K_SIZE*MODES_CLSNO>,R0 ; Compute offset to MODES CDB  
50 00000000'EF40 9E OFAB 3975 MOVAB CDBHEAD[R0],R0 ; ... get its CDB address  
52 2E A0 DO OFB3 3976 MOVL CDB$A_BUFFERS(R0),R2 ; ... and MBP ptr for later use  
OFA4 3977 :  
50 00000000'8F DO OFB7 3978 MOVL #<CDB$K_SIZE*STATES_CLSNO>,R0 ; Compute offset to STATES CDB  
50 00000000'EF40 9E OFBE 3979 MOVAB CDBHEAD[R0],R0 ; ... get its CDB address  
54 2E A0 DO OFC6 3980 MOVL CDB$A_BUFFERS(R0),R4 ; ... and MBP ptr for later use  
OFA4 3981 :  
55 2E A6 DO OFCA 3982 MOVL CDB$A_BUFFERS(R6),R5 ; Get same for SYSTEM class  
OFA4 3983 :  
58 42 A6 9A OFCE 3984 MCVZBL CDB$B_ST(R6),R8 ; Get requested stat code  
OFD2 3985 CASE R8,<FS_ALL,FS_CUR,FS_AVE,FS_MIN,FS_MAX>,L  
OFA4 3986 : ; Select on requested stat  
OFA4 3987 :  
OFA4 3988 : FS_ALL: ; Should not occur  
OFA4 3989 : FS_CUR:  
52 08 A2 DO OFE0 3990 MOVL MBP$A_STATS(R2),R2 ; Load addr of STATS buffer for MODES  
54 08 A4 DO OFE4 3991 MOVL MBP$A_STATS(R4),R4 ; Load addr of STATS buffer for STATES  
55 08 A5 DO OFE8 3992 MOVL MBP$A_STATS(R5),R5 ; Load addr of STATS buffer for SYSTEM  
2A 11 OFEC 3993 BRB FS_COMMON ; Join common code  
OFA4 3994 :  
OFA4 3995 : FS_AVE:  
52 14 A2 DO OFEE 3996 MOVL MBP$A_SUM(R2),R2 ; Load addr of SUM buffer for MODES  
54 14 A4 DO OFF2 3997 MOVL MBP$A_SUM(R4),R4 ; Load addr of SUM buffer for STATES  
55 14 A5 DO OFF6 3998 MOVL MBP$A_SUM(R5),R5 ; Load addr of SUM buffer for SYSTEM  
1C 11 OFFA 3999 BRB FS_COMMON ; Join common code  
OFA4 4000 :  
OFA4 4001 : FS_MIN:  
52 0C A2 DO OFFC 4002 MOVL MBP$A_MIN(R2),R2 ; Load addr of MIN buffer for MODES  
54 0C A4 DO 1000 4003 MOVL MBP$A_MIN(R4),R4 ; Load addr of MIN buffer for STATES  
55 0C A5 DO 1004 4004 MOVL MBP$A_MIN(R5),R5 ; Load addr of MIN buffer for SYSTEM  
OE 11 1008 4005 BRB FS_COMMON ; Join common code  
OFA4 4006 :  
OFA4 4007 : FS_MAX:  
52 10 A2 DO 100A 4008 MOVL MBP$A_MAX(R2),R2 ; Load addr of MAX buffer for MODES
```



```
54 10 A4 D0 100E 4009      MOVL    MBP$A_MAX(R4),R4      ; Load addr of MAX buffer for STATES
55 10 A5 D0 1012 4010      MOVL    MBP$A_MAX(R5),R5      ; Load addr of MAX buffer for SYSTEM
    00 11 1016 4011      BRB      FS_COMMON              ; Join common code
    1018 4012
    1018 4013 FS_COMMON:
    1018 4014
    1018 4015
    1018 4016 : Move items of interest from the source buffers to a single
    1018 4017 : destination buffer.
    1018 4018
    1018 4019 *** NOTE *** This section contains hard-wired offsets. It
    1018 4020 : assumes that the positions of items in the
    1018 4021 : MODES and STATES class do not change.
    1018 4022 : Not currently using the MODES class for the
    1018 4023 : single display, however.
    1018 4024
    1018 4025
    1018 4026
    1018 4027 *** NOTE *** R2 has the address of the MODES buffer, but it is
    1018 4028 : not being used currently.
    1018 4029
    1018 4030
    1018 4031
    57 53 D0 1018 4032      MOVL    R3,R7                ; Set up variable ptr to dummy buffer
    101B 4033
    87 85 D0 101B 4034      MOVL    (R5)+,(R7)+            ; Get "CPU busy" from SYSTEM
    87 10 A4 7D 101E 4035      MOVQ   16(R4),(R7)+        ; Get 8 items from STATES
    87 18 A4 7D 1022 4036      MOVQ   24(R4),(R7)+
    87 2C A4 7D 1026 4037      MOVQ   44(R4),(R7)+
    87 0C A4 D0 102A 4038      MOVL    12(R4),(R7)+
    87 04 A4 D0 102E 4039      MOVL    4(R4),(R7)+
    1032 4040
    50 FFFFFFFF7'8F D0 1032 4041      MOVL    #<ECOUNT_SYS_SINGLE-9>,R0 ; No. of items to get from SYSTEM
    1039 4042 10$:
    87 85 D0 1039 4043      MOVL    (R5)+,(R7)+            ; Move a SYSTEM item into dummy buffer
    FA 50 F5 103C 4044      SOBGTR  R0,10$                ; Loop to get all of them
    103F 4045
    103F 4046
```



```
103F 4048 :  
103F 4049 : At this point,  
103F 4050 :  
103F 4051 : R3 - points to the newly formed dummy statistics  
103F 4052 : R6 - CDB ptr  
103F 4053 : R8 - statistic code  
103F 4054 : R11 - TM4 ptr  
103F 4055 :  
103F 4056 :  
55 00000103'EF DE 103F 4057 MOVAL FAOSTK,R5 ; Load address of display buffer  
1046 4058 :  
1046 4059 :  
1046 4060 : Loop once for each element in this class. Pick up transformed value for  
1046 4061 : desired statistic from appropriate buffer within collection buffer  
1046 4062 : block. Do computation on transformed value if required, and place  
1046 4063 : whole and fractional portions of result into display buffer (FAOSTK).  
1046 4064 :  
1046 4065 :  
6B FFFFFFFF'8F D0 1046 4066 MOVL #<ECOUNT_SYS_SINGLE-1>, - ; Set up number of elements to  
104D 4067 TM4$$_ECOUNT(R11) ; ... display as a loop limit  
104D 4068 :  
50 00000000'EF DE 104D 4069 MOVAL ITMSTR_SYS_SINGLE,R0 ; Address of item-number string  
1054 4070 :  
52 D4 1054 4071 CLRL R2 ; Clear element index register  
1056 4072 FSS_BEG:  
54 80 9A 1056 4073 MOVZBL (R0)+,R4 ; Get next item number  
54 11 C4 1059 4074 MULL2 #IDB$K_ILENGTH,R4 ; Compute index into IDB table  
54 0000'CF44 9E 105C 4075 MOVAB W^PERTABLE[R4],R4 ; Address of IDB for this item  
00E0'CF 0A A4 B0 1062 4076 MOVW IDB$W_TYPE(R4),W^ITEM_TYPE ; Save item type for use below  
1068 4077 :  
02 58 D1 1068 4078 CMPL R8,#AVE_STAT ; AVERAGE statistic requested ?  
05 13 106B 4079 BEQL 10$ ; Br if yes  
FE83 30 106D 4080 BSBW INTORFL ; Process an integer or floating value  
03 11 1070 4081 BRB 20$ ; ... and go loop for next element  
1072 4082 10$:  
FE11 30 1072 4083 BSBW INTAVE ; Process an integer average value  
1075 4084 20$:  
FFDB 52 01 6B F1 1075 4085 ACBL TM4$$_ECOUNT(R11),#1,R2,FSS_BEG ; Loop once for each element  
107B 4086 :  
56 DD 107B 4087 PUSHL R6 ; Pass CDB address  
55 DD 107D 4088 PUSHL R5 ; Pass current FAOSTK address  
00001089'EF 02 FB 107F 4089 CALLS #2,FDB_SYS_TOP ; Compute and stack the four tops  
1086 4090 :  
FDE2 31 1086 4091 BRW FDB_RET ; Go return  
1089 4092 :
```

```

1089 4094      .SBTTL FDB_SYS_TOP - Process TOPs for SYSTEM class
1089 4095
1089 4096 :++
1089 4097 :
1089 4098 : FUNCTIONAL DESCRIPTION:
1089 4099 :
1089 4100 :     TBS
1089 4101 :
1089 4102 : INPUTS:
1089 4103 :
1089 4104 :
1089 4105 : OUTPUTS:
1089 4106 :
1089 4107 :
1089 4108 : IMPLICIT OUTPUTS:
1089 4109 :
1089 4110 :
1089 4111 : ROUTINE VALUE:
1089 4112 :
1089 4113 :     R0 = SSS_NORMAL
1089 4114 :
1089 4115 : SIDE EFFECTS:
1089 4116 :
1089 4117 :     none
1089 4118 :
1089 4119 :--
1089 4120 :
OFFC 1089 4121 .ENTRY FDB_SYS_TOP, *M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>

```



```
57 00000000'8F DO 108B 4123
50 00000000'EF47 9E 108B 4124      MOVL #<PROCS_CLSNO*CDB$K_SIZE>,R7 ; Compute offset to PROCESSES CDB
                                1092 4125      MOVAB CDBHEAD[R7],R0 ; Index to CDB address
                                109A 4126      ; NOTE -- TOP_DIFFS needs R0 set up this way
                                109A 4127      MOVL CDB$A_BUFFERS(R0),R7 ; Load buffer block ptr
                                109E 4128      MOVL MBP$A_BUFFERA(R7),R7 ; Load collection buffer ptr
                                10A1 4129
                                10A1 4130
                                10A1 4131 : If not first TOP display event, calculate S_TOP_TICKS (the
                                10A1 4132 : number of clock ticks (10ms units) since previous entry
                                10A1 4133 : to FDB_SYS_TOP.
                                10A1 4134
                                10A1 4135
52 00000000'EF DO 10A1 4136      MOVL MCAPTR,R2 ; Get MCA pointer
19 32 A2 09 E1 10A8 4137      BBC #MCA$V_S_TOP_DISP,MCA$W_FLAGS(R2),5$
                                10AD 4138      ; Skip TOP_TICKS calc if 1st time thru
52 03 A7 7D 10AD 4139      MOVQ MNR_CLS$Q_STAMP(R7),R2 ; Current system time to temp regs
52 002F'CF C2 10B1 4140      SUBL2 W^S_TOP_TIME,R2 ; Calc low-order in sys units
53 0033'CF D9 10B6 4141      SBWC W^S_TOP_TIME+4,R3 ; Calc high-order in sys units
0037'CF 52 000186A0 8F 7B 10BB 4142      EDIV #100000,R2,W^S_TOP_TICKS,R2 ; Calc interval ticks (10ms units)
52 10C5
                                10C6 4143      ; ... for use below
002F'CF 03 A7 7D 10C6 4144 5$: MOVQ MNR_CLS$Q_STAMP(R7),W^S_TOP_TIME ; Save curr time for next disp even
                                10CC 4145
                                10CC 4146
                                10CC 4147 : Set up registers to call TOP_DIFFS routine to get the DIFF array
                                10CC 4148 : filled in, which contains the incremental differences in the
                                10CC 4149 : requested resource since previous display event. R0 and R7
                                10CC 4150 : already set up.
                                10CC 4151
                                10CC 4152
                                10CC 4153
54 0000003B'EF DE 10CC 4154      MOVAL SYS_TOP_VEC,R4 ; Get ptr to vector of top buffers
52 84 DO 10D3 4155      MOVL (R4)+,R2 ; Get DATA array ptr
53 84 DO 10D6 4156      MOVL (R4)+,R3 ; Get DIFF array ptr
55 84 DO 10D9 4157      MOVL (R4)+,R5 ; Get PID array ptr
56 84 DO 10DC 4158      MOVL (R4)+,R6 ; Get ADDR array ptr
5B 2B DO 10DF 4159      MOVL #MNR_PROSL_CPUTIM,R11 ; Get CPUTIM offset
0081 8F BB 10E2 4160      PUSHF #^M<R0,R7> ; Save since changed by TOP_DIFFS
0304 30 10E6 4161      BSBW TOP_DIFFS ; Calc diffs over the last interval
0081 8F BA 10E9 4162      POPR #^M<R0,R7> ; Restore
                                10ED 4163
52 84 DO 10ED 4164      MOVL (R4)+,R2 ; Get DATA array ptr
53 84 DO 10F0 4165      MOVL (R4)+,R3 ; Get DIFF array ptr
55 84 DO 10F3 4166      MOVL (R4)+,R5 ; Get PID array ptr
56 84 DO 10F6 4167      MOVL (R4)+,R6 ; Get ADDR array ptr
5B 27 DO 10F9 4168      MOVL #MNR_PROSL_PAGEFLTS,R11 ; Get PAGEFLTS offset
0081 8F BB 10FC 4169      PUSHF #^M<R0,R7> ; Save since changed by TOP_DIFFS
02EA 30 1100 4170      BSBW TOP_DIFFS ; Calc diffs over the last interval
0081 8F BA 1103 4171      POPR #^M<R0,R7> ; Restore
                                1107 4172
52 84 DO 1107 4173      MOVL (R4)+,R2 ; Get DATA array ptr
53 84 DO 110A 4174      MOVL (R4)+,R3 ; Get DIFF array ptr
55 84 DO 110D 4175      MOVL (R4)+,R5 ; Get PID array ptr
56 84 DO 1110 4176      MOVL (R4)+,R6 ; Get ADDR array ptr
5B 23 DO 1113 4177      MOVL #MNR_PROSL_DIOCNT,R11 ; Get DIOCNT offset
0081 8F BB 1116 4178      PUSHF #^M<R0,R7> ; Save since changed by TOP_DIFFS
```



```
02D0 30 111A 4179 BSBW TOP_DIFFS ; Calc diffs over the last interval
0081 8F BA 111D 4180 POPR #^M<R0,R7> ; Restore
      1121 4181
52 84 D0 1121 4182 MOVL (R4)+,R2 ; Get DATA array ptr
53 84 D0 1124 4183 MOVL (R4)+,R3 ; Get DIFF array ptr
55 84 D0 1127 4184 MOVL (R4)+,R5 ; Get PID array ptr
56 84 D0 112A 4185 MOVL (R4)+,R6 ; Get ADDR array ptr
5B 2F D0 112D 4186 MOVL #MNR_PROSL_BIOCNT,R11 ; Get BIOCNT offset
0081 8F BB 1130 4187 PUSHF #^M<R0,R7> ; Save since changed by TOP_DIFFS
02B6 30 1134 4188 BSBW TOP_DIFFS ; Calc diffs over the last interval
0081 8F BA 1137 4189 POPR #^M<R0,R7> ; Restore
      113B 4190
      113B 4191 ; Now all DIFF arrays are established. Loop through each one to find
      113B 4192 ; the top user. Note: TOP_DIFFS places MAXPROCESSCNT in R11.
      113B 4193
      113B 4194
      113B 4195
55 04 AC D0 113B 4196 MOVL 4(AP),R5 ; Get current position in FAOSTK
56 08 AC D0 113F 4197 MOVL 8(AP),R6 ; Get CDB address for later use
54 0000003B'EF DE 1143 4198 MOVAL SYS_TOP_VEC,R4 ; Get addr of TOP arrays
51 00000000'EF D0 114A 4199 MOVL MCAPTR,R1 ; Get pointer to MCA
OA 32 A1 09 E2 1151 4200 BBSS #MCASV_S_TOP_DISP,MCASW_FLAGS(R1),10$ ; Br if not first disp
      1156 4201 ; ... and always set bit
      40 10 1156 4202 BSBB CLEAR_STACK ; If first display event, simply
      3E 10 1158 4203 BSBB CLEAR_STACK ; ... stack zeroes
      3C 10 115A 4204 BSBB CLEAR_STACK ; ...
      3A 10 115C 4205 BSBB CLEAR_STACK ; ...
      115E 4206
      37 11 115E 4207 BRB 20$ ; ... and go return
      1160 4208 10$:
51 04 A4 D0 1160 4209 MOVL 4(R4),R1 ; Get addr of CPUTIM DIFF array
53 0C A4 D0 1164 4210 MOVL 12(R4),R3 ; ... and ADDR array
      52 D4 1168 4211 CLRL R2 ; ... and set up index for CALC_BAR
      0054 30 116A 4212 BSBW STACK_TOP ; Find the top and stack its info
51 14 A4 D0 116D 4213 MOVL 20(R4),R1 ; Get addr of PAGEFLTS DIFF array
53 1C A4 D0 1171 4214 MOVL 28(R4),R3 ; ... and ADDR array
      52 0B D0 1175 4215 MOVL #11,R2 ; ... and set up index for CALC_BAR
      0046 30 1178 4216 BSBW STACK_TOP ; Find the top and stack its info
51 24 A4 D0 117B 4217 MOVL 36(R4),R1 ; Get addr of DIRIO DIFF array
53 2C A4 D0 117F 4218 MOVL 44(R4),R3 ; ... and ADDR array
      52 0F D0 1183 4219 MOVL #15,R2 ; ... and set up index for CALC_BAR
      0038 30 1186 4220 BSBW STACK_TOP ; Find the top and stack its info
51 34 A4 D0 1189 4221 MOVL 52(R4),R1 ; Get addr of BUFIO DIFF array
53 3C A4 D0 118D 4222 MOVL 60(R4),R3 ; ... and ADDR array
      52 10 D0 1191 4223 MOVL #16,R2 ; ... and set up index for CALC_BAR
      002A 30 1194 4224 BSBW STACK_TOP ; Find the top and stack its info
      1197 4225 20$:
      04 1197 4226 RET ; Return
      1198 4227
      1198 4228 CLEAR_STACK:
      85 7C 1198 4229 CLRQ (R5)+ ; Stack null process name
      85 7C 119A 4230 CLRQ (R5)+ ; ...
      85 FO A5 9A 119C 4231 MOVZBL -16(R5),(R5)+ ; Length of process name to FAO stack
85 55 13 C3 11A0 4232 SUBL3 #19,R5,(R5)+ ; Address of process name to FAO stack
      85 7C 11A4 4233 CLRQ (R5)+ ; Stack dummy value and bar value
      05 11A6 4234 RSB
      11A7 4235
```



```
11A7 4236 FIND_TOP:
5A 50 D4 11A7 4237 CLRL R0 ; Init index
6140 58 D0 11A9 4238 MOVL (R1)[R0],R10 ; Get first value
58 D4 11AD 4239 CLRL R8 ; ... and first index
6140 5A D1 11AF 4240 10$: CMPL R10,(R1)[R0] ; This item greater than current best?
07 18 11B3 4241 BGEQ 20$ ; Br if no
5A 6140 D0 11B5 4242 MOVL (R1)[R0],R10 ; Found a new best
58 50 D0 11B9 4243 MOVL R0,R8 ; Save its index
EF 50 5B F2 11BC 4244 20$: AOBLS R11,R0,10$ ; Loop MAXPROCESSCNT times
05 11C0 4245 RSB ; Return
11C1 4246
11C1 4247
11C1 4248 STACK_TOP:
E4 10 11C1 4249 BSBB FIND_TOP ; Find the top process
11C3 4250
11C3 4251 ; At this point, R10 has the top value, and R8 has its index value.
11C3 4252
11C3 4253
11C3 4254
11C3 4255
11C3 4256
50 08 A4 D0 11C3 4257 MOVL 8(R4),R0 ; Get addr of CPUTIM PID array
50 6048 D0 11C7 4258 MOVL (R0)[R8],R0 ; Get PID for top process
50 B5 11CB 4259 TSTW R0 ; Is it the NULL process?
05 12 11CD 4260 BNEQ 10$ ; Br if no
6148 D4 11CF 4261 CLRL (R1)[R8] ; NULL process -- zero its DIFF val
D3 10 11D2 4262 BSBB FIND_TOP ; ... and go find another top
11D4 4263 10$: TSTL R10 ; Top value zero ?
5A D5 11D4 4264 BNEQ 30$ ; Br if no
04 12 11D6 4265 BSBB CLEAR_STACK ; Yes, simply stack zeroes
BE 10 11D8 4266 BRB 40$ ; ... and go exit
2C 11 11DA 4267
11DC 4268 30$:
58 6348 D0 11DC 4269 MOVL (R3)[R8],R8 ; Get addr of process data block
85 0B A8 7D 11E0 4270 MOVQ MNR_PRO$0_LNAME(R8),(R5)+ ; Process name cstring to FAO stack
85 13 A8 7D 11E4 4271 MOVQ MNR_PRO$0_LNAME+8(R8),(R5)+ ; Length of process name to FAO stack
85 F0 A5 9A 11E8 4272 MOVZBL -16(R5),(R5)+ ; Address of process name to FAO stack
85 55 13 C3 11EC 4273
11F0 4274
11F0 4275 ; Transform the DIFF value (in R10) from a delta to a
11F0 4276 ; rate/second. Then place it into the FAOSTK and ship
11F0 4277 ; it off to the CALC_BAR subroutine to insert the
11F0 4278 ; bar character count into FAOSTK.
11F0 4279
11F0 4280
11F0 4281
11F0 4282 CVTLF R10,R10 ; Get floating value over interval
50 5A 5A 4E 11F0 4282 CVTLF W^S_TOP-TICKS,R0 ; Get floating ticks over interval
50 000043C8 8F 4E 11F3 4283 DIVF2 #100,R0 ; Get floating seconds over interval
5A 50 46 11F8 4284 DIVF2 R0,R10 ; Get floating rate per second
85 5A 46 11FF 4285 CVTFL R10,(R5)+ ; Move longword rate/sec to FAO stack
FD26 30 1205 4286 BSBW CALC_BAR ; Calculate and stack bar chars required
1208 4287 ; CALC_BAR destroys regs R7 and R9, and
1208 4288 ; ... updates R5 to point to the next
1208 4289 ; ... available longword in the FAO stack
1208 4290
1208 4291 40$:
05 1208 4292 RSB ; Return
```

MONITOR
V04-000

- VAX/VMS Performance Monitor Utility^{1 4}
FDB_SYS_TOP - Process TOPs for SYSTEM cl 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 Page 106
1209 4293 (58)


```
1209 4295 :  
1209 4296 : FDB_SYS_ALL  
1209 4297 :  
1209 4298 : Fill a dummy MBP and 4 dummy statistics buffers (STATS, MIN, MAX, and SUM).  
1209 4299 : The dummy statistics buffers will form a hybrid between the MODES class  
1209 4300 : and the SYSTEM class.  
1209 4301 :  
1209 4302 : Inputs:  
1209 4303 :  
1209 4304 : R0 - R5 scratch  
1209 4305 : R6 - address of SYSTEM CDB.  
1209 4306 : R10 - address of dummy MBP followed by the 4 statistics buffers.  
1209 4307 : R11 - address of TM4, a temporary stack area  
1209 4308 :  
1209 4309 : Implicit Outputs:  
1209 4310 :  
1209 4311 : The dummy MBP and buffers are filled.  
1209 4312 : TM4$E_COUNT contains number of elements to display.  
1209 4313 : TM4$A_ITMSTR points to the display item string.  
1209 4314 :  
1209 4315 :  
1209 4316 : FDB_SYS_ALL:  
1209 4317 :  
50 00000000'8F DO 1209 4318 MOVL #<CDB$K_SIZE*MODES_CLSNO>,R0 ; Compute offset to MODES CDB  
50 00000000'EF40 9E 1210 4319 MOVAB CDBHEAD[R0],R0 ; ... get its CDB address  
51 2E A0 DO 1218 4320 MOVL CDB$A_BUFFERS(R0),R1 ; ... and MBP ptr for later use  
52 2E A6 DO 121C 4321 MOVL CDB$A_BUFFERS(R6),R2 ; Get same for SYSTEM class  
53 5A 28 C1 1220 4322  
1224 4323 ADDL3 #MBP$K_SIZE,R10,R3 ; Compute address of dummy STATS buffer  
1224 4324 :  
1224 4325 : Fill STATS buffer with E_COUNT_SYS_ALL longwords  
1224 4326 : starting at address in R3.  
1224 4327 :  
1224 4328 :  
1224 4329 :  
08 AA 53 DO 1224 4330 MOVL R3,MBP$A_STATS(R10) ; Load addr of dummy STATS buffer  
54 08 A1 DO 1228 4331 MOVL MBP$A_STATS(R1),R4 ; Get MODES STATS buffer ptr  
55 07 DO 122C 4332 MOVL #7,R5 ; ... and no. of items to pick up  
6F 10 122F 4333 BSBB MOVE_ITEMS ; Move the items into the dummy buffer  
1231 4334 :  
54 08 A2 08 C1 1231 4335 ADDL3 #8,MBP$A_STATS(R2),R4 ; Get SYSTEM STATS buff ptr (skips 2 items)  
55 FFFFFFFF'8F DO 1236 4336 MOVL #<E_COUNT_SYS_ALL-7>,R5 ; ... and no. of items to pick up  
61 10 123D 4337 BSBB MOVE_ITEMS ; Move the items into the dummy buffer  
123F 4338 :  
123F 4339 : Fill MIN buffer with E_COUNT_SYS_ALL longwords  
123F 4340 : starting at address in R3.  
123F 4341 :  
123F 4342 :  
123F 4343 :  
0C AA 53 DO 123F 4344 MOVL R3,MBP$A_MIN(R10) ; Load addr of dummy MIN buffer  
54 0C A1 DO 1243 4345 MOVL MBP$A_MIN(R1),R4 ; Get MODES MIN buffer ptr  
55 07 DO 1247 4346 MOVL #7,R5 ; ... and no. of items to pick up  
54 10 124A 4347 BSBB MOVE_ITEMS ; Move the items into the dummy buffer  
124C 4348 :  
54 0C A2 08 C1 124C 4349 ADDL3 #8,MBP$A_MIN(R2),R4 ; Get SYSTEM MIN buff ptr (skips 2 items)  
55 FFFFFFFF'8F DO 1251 4350 MOVL #<E_COUNT_SYS_ALL-7>,R5 ; ... and no. of items to pick up  
46 10 1258 4351 BSBB MOVE_ITEMS ; Move the items into the dummy buffer
```

```
125A 4352
125A 4353 :
125A 4354 : Fill MAX buffer with ECOUNT_SYS_ALL longwords
125A 4355 : starting at address in R3.
125A 4356 :
125A 4357 :
10 AA 53 D0 125A 4358 MOVL R3,MBP$A_MAX(R10) : Load addr of dummy MAX buffer
54 10 A1 D0 125E 4359 MOVL MBP$A_MAX(R1),R4 : Get MODES MAX buffer ptr
55 07 D0 1262 4360 MOVL #7,R5 : ... and no. of items to pick up
39 10 1265 4361 BSBB MOVE_ITEMS : Move the items into the dummy buffer
1267 4362 :
54 10 A2 08 C1 1267 4363 ADDL3 #8,MBP$A_MAX(R2),R4 : Get SYSTEM MAX buff ptr (skips 2 items)
55 FFFFFFFF'8F D0 126C 4364 MOVL #<ECOUNT_SYS_ALL-7>,R5 : ... and no. of items to pick up
2B 10 1273 4365 BSBB MOVE_ITEMS : Move the items into the dummy buffer
1275 4366 :
1275 4367 :
1275 4368 : Fill SUM buffer with ECOUNT_SYS_ALL longwords
1275 4369 : starting at address in R3.
1275 4370 :
1275 4371 :
14 AA 53 D0 1275 4372 MOVL R3,MBP$A_SUM(R10) : Load addr of dummy SUM buffer
54 14 A1 D0 1279 4373 MOVL MBP$A_SUM(R1),R4 : Get MODES SUM buffer ptr
55 07 D0 127D 4374 MOVL #7,R5 : ... and no. of items to pick up
1E 10 1280 4375 BSBB MOVE_ITEMS : Move the items into the dummy buffer
1282 4376 :
54 14 A2 08 C1 1282 4377 ADDL3 #8,MBP$A_SUM(R2),R4 : Get SYSTEM SUM buff ptr (skips 2 items)
55 FFFFFFFF'8F D0 1287 4378 MOVL #<ECOUNT_SYS_ALL-7>,R5 : ... and no. of items to pick up
10 10 128E 4379 BSBB MOVE_ITEMS : Move the items into the dummy buffer
1290 4380 :
1290 4381 :
6B 00000000'8F D0 1290 4382 MOVL #ECOUNT_SYS_ALL, - : Load number of elements to display
04 AB 00000000'EF DE 1297 4383 TM4$E ECOUNT(R11)
129F 4384 MOVAL ITMSTR_SYS_ALL, -
129F 4385 TM4$A_ITMSTR(R11) : ... and addr of display item string
129F 4386 :
05 129F 4387 RSB : Return to caller
12A0 4388 :
12A0 4389 :
12A0 4390 :
12A0 4391 : MOVE_ITEMS moves a consecutive number of longwords (number in R5)
12A0 4392 : from location R4 to location R3. R4 and R3 are auto-incremented.
12A0 4393 :
12A0 4394 :
12A0 4395 MOVE_ITEMS:
12A0 4396 10$:
83 84 D0 12A0 4397 MOVL (R4)+,(R3)+ : Move an item into dummy buffer
FA 55 F5 12A3 4398 SOBGTR R5,10$ : Loop to get all of them
05 12A6 4399 RSB
12A7 4400
```



```
12A7 4402      .SBTTL FILL_TOP - Fill Display Buffer for TOP PROCESSES
12A7 4403
12A7 4404      :++
12A7 4405      :
12A7 4406      : FUNCTIONAL DESCRIPTION:
12A7 4407      :
12A7 4408      :     Calculates the TOP 8 PROCESSES since the last display
12A7 4409      :     event, and fills the display buffer (FAOSTK) with data
12A7 4410      :     for later display.
12A7 4411      :
12A7 4412      : INPUTS:
12A7 4413      :
12A7 4414      :     4(AP) - CDB (Class Descriptor Block) pointer
12A7 4415      :     for the PROCESSES class.
12A7 4416      :
12A7 4417      : IMPLICIT INPUTS:
12A7 4418      :
12A7 4419      :     FAOSTK - FAO parameter list for a TOP screen
12A7 4420      :     MCAPTR - Pointer to MCA (Monitor Communication Area)
12A7 4421      :
12A7 4422      : OUTPUTS:
12A7 4423      :
12A7 4424      :     None
12A7 4425      :
12A7 4426      : IMPLICIT OUTPUTS:
12A7 4427      :
12A7 4428      :     Entire display buffer (FAOSTK) filled with data for
12A7 4429      :     eventual display.
12A7 4430      :
12A7 4431      :     TOP_PROCS byte filled with number of TOP processes
12A7 4432      :     (with non-zero values) to display.
12A7 4433      :
12A7 4434      : ROUTINE VALUE:
12A7 4435      :
12A7 4436      :     R0 = $$$_NORMAL
12A7 4437      :
12A7 4438      : SIDE EFFECTS:
12A7 4439      :
12A7 4440      :     none
12A7 4441      :
12A7 4442      : REGISTER USAGE:
12A7 4443      :
12A7 4444      :     R0 = Scratch
12A7 4445      :     R1 = Scratch, addr of MBP
12A7 4446      :     R2 = Addr of DATA array
12A7 4447      :     R3 = Addr of DIFF array
12A7 4448      :     R4 = Addr of ORDER array
12A7 4449      :     R5 = Addr of PID array
12A7 4450      :     R6 = Addr of ADDR array
12A7 4451      :     R7 = Pointer to collection buffer data block
12A7 4452      :     R8 = Process index (PIX) for current data block
12A7 4453      :     R9 = Current process index (from 0 to MNR_SYISW_MAXPRCCT)
12A7 4454      :     R10 = Number of data blocks (processes) in coll buff
12A7 4455      :     R11 = Pointer to monitored data item in coll buff data block;
12A7 4456      :           Also, Max process count (from MNR_SYISW_MAXPRCCT)
12A7 4457      : --
12A7 4458
```

MONITOR
V04-000

M 4
- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
FILL_TOP - Fill Display Buffer for TOP P 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 Page 110
OFFC 12A7 4459 .ENTRY FILL_TOP, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> (60)


```
51 04 AC D0 12A9 4461      MOVL 4(AP),R1      ; Load CDB pointer
5B 42 A1 9A 12AD 4462      MOVZBL CDB$B_ST(R1),R11 ; Get PROCESSES display type code
                                CASE R11,<REG,TOPC,TPD,TPB,TPF>,W ; Go set offset based on type
                                BRB TOPC ; If out of range, do a TOPCPU
                                00 11 12BF 4464
                                12C1 4465
                                12C1 4466 REG: ; Regular PROCESSES display (should not get)
                                12C1 4467 TOPC: ; TOPCPU display
5B 2B D0 12C1 4468      MOVL #MNR_PROSL_CPUTIM,R11 ; Get offset into PROCESSES data block
0D 11 12C4 4469      BRB FT_CASE ; Join common code
                                12C6 4470 TOPD: ; TOPDIO display
5B 23 D0 12C6 4471      MOVL #MNR_PROSL_DIOCNT,R11 ; Get offset into PROCESSES data block
08 11 12C9 4472      BRB FT_CASE ; Join common code
                                12CB 4473 TOPB: ; TOPBIO display
5B 2F D0 12CB 4474      MOVL #MNR_PROSL_BIOCNT,R11 ; Get offset into PROCESSES data block
03 11 12CE 4475      BRB FT_CASE ; Join common code
                                12D0 4476 TOPF: ; TOPFAULT display
5B 27 D0 12D0 4477      MOVL #MNR_PROSL_PAGEFLTS,R11 ; Get offset into PROCESSES data block
                                12D3 4478
                                12D3 4479 FT_CASE: ; Common CASE return
00 11 12D3 4480
51 2E A1 D0 12D3 4481      MOVL CDB$A_BUFFERS(R1),R1 ; Load buffer block ptr
57 61 D0 12D7 4482      MOVL MBP$A_BUFFERA(R1),R7 ; Load collection buffer ptr
                                12DA 4483
                                12DA 4484 ;
                                12DA 4485 ; If not first TOP display event, calculate TOP TICKS (the
                                12DA 4486 ; number of clock ticks (10ms units) since previous entry
                                12DA 4487 ; to FILL_TOP.
                                12DA 4488 ;
                                12DA 4489
52 00000000'EF D0 12DA 4490      MOVL MCAPTR,R2 ; Get MCA pointer
19 32 A2 07 E1 12E1 4491      BBC #MCA$V_TOP_DISP,MCA$W_FLAGS(R2),5$
                                12E6 4492 ; Skip TOP TICKS calc if 1st time thru
52 03 A7 7D 12E6 4493      MOVQ MNR_CLSSQ_STAMP(R7),R2 ; Current system time to temp regs
52 0023'CF C2 12EA 4494      SUBL2 W^TOP_TIME,R2 ; Calc low-order in sys units
53 0027'CF D9 12EF 4495      SBWC W^TOP_TIME+4,R3 ; Calc high-order in sys units
002B'CF 52 000186A0 8F 7B 12F4 4496      EDIV #100000,R2,W^TOP_TICKS,R2 ; Calc interval ticks (10ms units)
                                12FE 4497
                                12FF 4498 ; ... for use in MOVE_TOP8 rtn below
0023'CF 03 A7 7D 12FF 4499      MOVQ MNR_CLSSQ_STAMP(R7),W^TOP_TIME ; Save curr time for next disp event
                                1305 4500
                                1305 4501 ;
                                1305 4502 ; Set up array pointers in preparation for calculations of difference
                                1305 4503 ; values for each process for the monitored item over the last interval.
                                1305 4504 ;
                                1305 4505
52 08 A1 D0 1305 4506      MOVL MBP$A_DATA(R1),R2 ; Load DATA array ptr
53 0C A1 D0 1309 4507      MOVL MBP$A_DIFF(R1),R3 ; Load DIFF array ptr
54 10 A1 D0 130D 4508      MOVL MBP$A_ORDER(R1),R4 ; Load ORDER array ptr
55 14 A1 D0 1311 4509      MOVL MBP$A_PID(R1),R5 ; Load PID array ptr
56 18 A1 D0 1315 4510      MOVL MBP$A_ADDR(R1),R6 ; Load ADDR array ptr
50 04 AC D0 1319 4511      MOVL 4(AP),R0 ; Load CDB pointer
                                131D 4512
                                00CD 30 131D 4513      BSBW TOP_DIFFS ; Calculate the diffs over last int
                                1320 4514
51 00000000'EF D0 1320 4515      MOVL MCAPTR,R1 ; Get pointer to MCA
0E 32 A1 07 E3 1327 4516      BBCS #MCA$V_TOP_DISP,MCA$W_FLAGS(R1),80$
```

- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00 Page 112
 FILL_TOP - Fill Display Buffer for TOP P 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 (61)

			132C	4517			; If first top display event, don't
			132C	4518			; ... sort or move (and set bit for future)
0800	8F	BB	132C	4519	PUSHR	#^M<R11>	; Save max process count
	14	10	1330	4520			
			1330	4521	BSBB	SORT_PROCS	; Sort the top 8 processes
			1332	4522			
0800	8F	BA	1332	4523	POPR	#^M<R11>	; Restore max process count
			1336	4524			
	4B	10	1336	4525	BSBB	MOVE_TOP8	; Insert data for TOP 8 into FAOSTK
			1338	4526			
	04	11	1338	4527	BRB	90\$	; Go return
			133A	4528			
	0022'CF	94	133A	4529	CLRB	W^TOP_PROCS	; No procs to display on 1st time
			133E	4530			
50	00000000'EF	D0	133E	4531	MOVL	NORMAL,R0	; Indicate normal status
		04	1345	4532	RET		; Return


```
1346 4534 :  
1346 4535 : SORT_PROCS  
1346 4536 :  
1346 4537 : Set up the ORDER array to contain the processes indices.  
1346 4538 :  
1346 4539 : R11 contains max process count.  
1346 4540 :  
1346 4541 : This subroutine destroys registers R0,R1,R7,R8,R9,R10,R11.  
1346 4542 :  
1346 4543 :  
1346 4544 SORT_PROCS:  
1346 4545 :  
6441 51 D4 1346 4546 : CLRL R1 ; Zero first process index  
F8 51 5B D0 1348 4547 10$: MOVL R1,(R4)[R1] ; Load index into corresponding ORDER elem  
5B F2 134C 4548 AOBLS R11,R1,10$ ; Do all elements of ORDER array  
1350 4549 :  
1350 4550 :  
1350 4551 : Go through the DIFF array and re-position elements in the  
1350 4552 : ORDER array using a bubble sort. When the following two-level  
1350 4553 : loop is complete, the highest-numbered 8 elements of the ORDER  
1350 4554 : array will contain the process index numbers of the TOP 8  
1350 4555 : consumers of the monitored resource.  
1350 4556 :  
1350 4557 :  
5A 5B 5B D7 1350 4558 : DECL R11 ; Get highest process index (PIX)  
C3 1352 4559 : SUBL3 #7,R11,R10 ; Get 8th from the highest PIX  
59 01 D0 1356 4560 20$: MOVL #1,R9 ; Init loop index of inner loop  
51 59 01 C3 1359 4562 30$: SUBL3 #1,R9,R1 ; R1 is always one less than R9  
57 6449 D0 135D 4564 : MOVL (R4)[R9],R7 ; Get PIX from current ORDER element  
58 6441 D0 1361 4565 : MOVL (R4)[R1],R8 ; Get PIX from previous ORDER element  
6348 6347 D1 1365 4566 : CMPL (R3)[R7],(R3)[R8] ; Compare curr DIFF val with previous  
08 18 136A 4567 : BGEQ 40$ ; Curr is not less -- no switching  
6449 58 D0 136C 4568 : MOVL R8,(R4)[R9] ; Curr is less -- switch PIX in current  
6441 57 D0 1370 4569 : MOVL R7,(R4)[R1] ; ... ORDER elt with that in prev ORDER elt  
E1 59 5B F3 1374 4570 40$: AOBLEQ R11,R9,30$ ; Loop through all elements of ORDER  
1378 4572 : ; ... array except the ones on the high  
1378 4573 : ; ... end which already have TOP values  
1378 4574 :  
FFD4 5B FFFFFFFF 8F 5A F1 1378 4575 : ACBL R10,#-1,R11,20$ ; Loop 8 times to 'bubble down' PIX's  
1382 4576 : ; ... for the 8 largest consumers  
1382 4577 :  
05 1382 4578 : RSB ; Return  
1383 4579 :
```

```
1383 4581 :  
1383 4582 : MOVE_TOP8  
1383 4583 :  
1383 4584 : Move data for the top 8 (or fewer) processes  
1383 4585 : into FAOSTK for later display.  
1383 4586 :  
1383 4587 : R11 contains max process count.  
1383 4588 :  
1383 4589 : This subroutine destroys registers R0,R1,R5,R7,R8,R9,R10,R11.  
1383 4590 :  
1383 4591 :  
1383 4592 : MOVE_TOP8:  
1383 4593 :  
55 0103'CF DE 1383 4594 MOVAL W^FAOSTK,R5 ; Get pointer to display buffer  
51 01 D0 1388 4595 MOVL #1,R1 ; Init count of procs that have  
1388 4596 : ... DIFF value > 0  
1388 4597 10$: : Beginning of TOP 8 loop  
58 5B D7 138B 4598 DECL R11 ; Point to next lower ORDER element  
58 644B D0 138D 4599 MOVL (R4)[R11],R8 ; Get PIX from ORDER array  
5A 634B D0 1391 4600 MOVL (R3)[R8],R10 ; Get DIFF value for this TOP process  
4F 13 1395 4601 BEQL 40$ ; Get out of loop if zero  
58 664B D0 1397 4602 MOVL (R6)[R8],R8 ; Get ptr to process data block  
139B 4603 : ... from ADDR array  
139B 4604 :  
139B 4605 : NOTE -- at this point, R8 points to process data block  
139B 4606 : ... and R10 has DIFF value.  
139B 4607 :  
139B 4608 :  
139B 4609 :  
139B 4610 : Move individual items for this process from current data block  
139B 4611 : in collection buffer to longwords in FAO stack.  
139B 4612 :  
139B 4613 :  
50 04 AC D0 139B 4614 MOVL 4(AP),R0 ; Get PROCESSES CDB address  
139F 4615 :  
20 A0 33 B1 139F 4616 CMPW #MNR_PROSL_EPID,CDB$W_BLKLEN(R0) ; See if we have an EPID  
05 19 13A3 4617 BLSS 20$ ; Br if we do  
85 68 D0 13A5 4618 MOVL MNR_PROSL_IPID(R8),(R5)+ ; Get the Internal PID  
04 11 13A8 4619 BRB 30$ ; Go get process name  
13AA 4620 20$:  
85 33 A8 D0 13AA 4621 MOVL MNR_PROSL_EPID(R8),(R5)+ ; Get the Extended PID  
13AE 4622 30$:  
85 0B A8 7D 13AE 4623 MOVQ MNR_PROSL_LNAME(R8),(R5)+ ; Process name cstring to FAO stack  
85 13 A8 7D 13B2 4624 MOVQ MNR_PROSL_LNAME+8(R8),(R5)+ ;  
85 F0 A5 9A 13B6 4625 MOVZBL -16(R5),(R5)+ ; Length of process name to FAO stack  
85 55 13 C3 13BA 4626 SUBL3 #19,R5,(R5)+ ; Address of process name to FAO stack  
13BE 4627 :  
13BE 4628 :  
13BE 4629 : Transform the DIFF value (in R10) from a delta to a  
13BE 4630 : rate/second. Then place it into the FAOSTK and ship  
13BE 4631 : it off to the CALC_BAR subroutine to insert the  
13BE 4632 : bar character count into FAOSTK.  
13BE 4633 :  
13BE 4634 :  
50 5A 5A 4E 13BE 4635 CVTLF R10,R10 ; Get floating value over interval  
50 002B'CF 4E 13C1 4636 CVTLF W^TOP_TICKS,R0 ; Get floating ticks over interval  
50 000043C8 8F 46 13C6 4637 DIVF2 #100,R0 ; Get floating seconds over interval
```


5A 50	46 13CD	4638	DIVF2	R0,R10	; Get floating rate per second
85 5A	4A 13D0	4639	CVTFL	R10,(R5)+	; Move longword rate/sec to FAO stack
0040 8F	BB 13D3	4640	PUSHR	#^M<R6>	; Save ADDR array pointer
56 04 AC	D0 13D7	4641	MOVL	4(AP),R6	; ... and set up CDB ptr for CALC_BAR
FB50	30 13DB	4642	BSBW	CALC_BAR	; Calculate and stack bar chars required
		13DE			; CALC_BAR destroys regs R7 and R9, and
		13DE			; ... updates R5 to point to the next
		13DE			; ... available longword in the FAO stack
0040 8F	BA 13DE	4646	POPR	#^M<R6>	; Restore ADDR array pointer
		13E2			
A5 51 08	F3 13E2	4648	AOBLEQ	#8,R1,10\$	; Get info from TOP 8 processes
		13E6			
0022'CF	51 01	83 13E6	SUBB3	#1,R1,W^TOP_PROCS	; Adjust count of processes for
					; ... DISPLAY_TOP routine
		13EC			
		13EC			
	05	13EC	RSB		; Return
		13ED			
		4654			

```
13ED 4656 TOP_DIFFS:
13ED 4657
13ED 4658 :
13ED 4659 : Fill the DIFF array and ADDR array for the current collection buffer.
13ED 4660 :
13ED 4661 :
13ED 4662 : REGISTER INPUTS:
13ED 4663 :
13ED 4664 : R0 = PROCESSES CDB ptr
13ED 4665 : R1 = scratch
13ED 4666 : R2 = DATA array ptr
13ED 4667 : R3 = DIFF array ptr
13ED 4668 : R5 = PID array ptr
13ED 4669 : R6 = ADDR array ptr
13ED 4670 : R7 = pointer to PROCESSES collection buffer
13ED 4671 : R8-10 = scratch
13ED 4672 : R11 = offset into PROCESSES data block for requested resource
13ED 4673 :
13ED 4674 :
5A 57 0D CO 13ED 4675 ADDL2 #MNR_CLSSK_HSIZE,R7 ; Point to PROCESSES class prefix
5A 04 A7 DO 13F0 4676 MOVL MNR_PROSL_PCTINT(R7),R10 ; Get no. of procs in this coll buffer
57 08 CO 13F4 4677 ADDL2 #MNR_PROSL_PSIZE,R7 ; Point to first data block
5B 57 CO 13F7 4678 ADDL2 R7,R11 ; ... and to first monitored data item
13FA 4679
59 FFFFFFFF 8F DO 13FA 4680 MOVL #-1,R9 ; Init process index
```



```

      1401 4682 10$:
58 67 3C 1401 4683 MOVZWL MNR_PROSL_IPID(R7),R8 ; Get process index from next process
      1404 4684 ; ... in collection buffer
      1404 4685 20$:
      59 D6 1404 4686 INCL R9 ; Get next process index
58 59 D1 1406 4687 CMPL R9,R8 ; Any process slots not in coll buff?
      05 18 1409 4688 BGEQ 30$ ; No -- go process this one
6349 D4 140B 4689 CLRL (R3)[R9] ; Yes -- clear DIFF array for this index
      F4 11 140E 4690 BRB 20$ ; Loop back to check next index
      1410 4691 30$:
67 6548 D1 1410 4692 CMPL (R5)[R8],MNR_PROSL_IPID(R7) ; Same process as last time?
      15 12 1414 4693 BNEQU 40$ ; No -- go zero out DIFF
      6B D5 1416 4694 TSTL (R11) ; Zero data item => swapped out
      11 13 1418 4695 BEQL 40$ ; Swapped out -- go zero out DIFF
6248 D5 141A 4696 TSTL (R2)[R8] ; Swapped out last time?
      0C 13 141D 4697 BEQL 40$ ; Yes -- go zero out DIFF
6348 6B 6248 C3 141F 4698 SUBL3 (R2)[R8],(R11),(R3)[R8] ; Calculate DIFF
6648 57 D0 1425 4699 MOVL R7,(R6)[R8] ; Store proc data block ptr in ADDR array
      07 11 1429 4700 BRB 50$ ; ... and continue
      142B 4701 40$:
6348 6348 D4 142B 4702 CLRL (R3)[R8] ; Clear DIFF, indicating not a TOP candidate
6548 67 D0 142E 4703 MOVL MNR_PROSL_IPID(R7),(R5)[R8] ; Store PID
      1432 4704 50$:
6248 6B D0 1432 4705 MOVL (R11),(R2)[R8] ; Store data item into DATA array
51 20 A0 3C 1436 4706 MOVZWL CDB$W_BLKLEN(R0),R1 ; Get size of a data block
      57 51 C0 143A 4707 ADDL2 R1,R7 ; Point to next process in coll buffer
      5B 51 C0 143D 4708 ADDL2 R1,R11 ; ... and to next data item
      BE 5A F5 1440 4709 SOBGTR R10,10$ ; Loop once for each proc in coll buffer
      1443 4710
      1443 4711 ;
      1443 4712 ; A DIFF entry has been made for every index up through
      1443 4713 ; the last process in the collection buffer.
      1443 4714 ; The following loop clears the DIFF entry for each
      1443 4715 ; index between the last one already done and the
      1443 4716 ; last one in the DIFF array.
      1443 4717 ;
      1443 4718
      1443 4719
5B 00000000'EF 58 D6 1443 4719 INCL R8 ; Get next process index
      5B 0B AB D0 1445 4720 MOVL SPTR,R11 ; Get ptr to System Info Area
      5B 58 3C 144C 4721 MOVZWL MNR_SYISW_MAXPRCCT(R11),R11 ; Get max process count
      07 18 1453 4722 CMPL R8,R11 ; Any more process slots?
      1455 4723 BGEQ 70$ ; No -- skip clear loop
      1455 4724 60$:
6348 D4 1455 4725 CLRL (R3)[R8] ; Clear DIFF, indicating not a TOP candidate
      1458 4726
F9 58 5B F2 1458 4727 AOBLSS R11,R8,60$ ; Continue to end of DIFF array
      145C 4728 70$:
      145C 4729
      05 145C 4730 RSB ; Return to caller
```

```
145D 4732      .SBTTL SUMMARY_TOP - Set up Summary for TOP
145D 4733
145D 4734      :++
145D 4735
145D 4736      : FUNCTIONAL DESCRIPTION:
145D 4737
145D 4738      : This routine is called to do setup for Summary processing of
145D 4739      : the SYSTEM or PROCESSES class with the TOP display format. It is not
145D 4740      : called when summarizing with the regular PROCESSES display format
145D 4741      : or the tabular SYSTEM display format.
145D 4742
145D 4743      : The basic job of this routine is to call FDB_SYS_TOP or FILL_DISP_BUFF
145D 4744      : with the first PROCESSES collection buffer of the MONITOR request.
145D 4745      : This is accomplished by loading the pointer to the current collection
145D 4746      : buffer with the first collection buffer pointer, and then doing
145D 4747      : a normal FDB_SYS_TOP or FILL_DISP_BUFF call. The current collection
145D 4748      : buffer pointer is then restored to its original value before returning
145D 4749      : to caller.
145D 4750
145D 4751      : CALLING SEQUENCE:
145D 4752
145D 4753      : CALLS #1,SUMMARY_TOP
145D 4754
145D 4755      : INPUTS:
145D 4756
145D 4757      : 4(AP) - address of a pointer to the CDB (Class Descriptor Block)
145D 4758      : for either the SYSTEM or the PROCESSES class.
145D 4759
145D 4760      : IMPLICIT INPUTS:
145D 4761
145D 4762      : MCAPTR - pointer to MCA (Monitor Communication Area)
145D 4763
145D 4764      : MRBPTR - pointer to MRB (Monitor Request Block)
145D 4765
145D 4766      : SPTR - pointer to SYI (System Information Area)
145D 4767
145D 4768      : PROCS_CLSNO - class number for the PROCESSES class
145D 4769
145D 4770      : SYS_TOP_VEC - vector of pointers to SYSTEM class TOP arrays
145D 4771
145D 4772      : OUTPUTS:
145D 4773
145D 4774      : None.
145D 4775
145D 4776      : IMPLICIT OUTPUTS:
145D 4777
145D 4778      : The 5 TOP arrays (DATA, DIFF, ORDER, PID, ADDR)
145D 4779      : are filled with information from the first collection
145D 4780      : buffer.
145D 4781
145D 4782      : ROUTINE VALUE:
145D 4783
145D 4784      : RO = $$$_NORMAL
145D 4785
145D 4786      : SIDE EFFECTS:
145D 4787
145D 4788      : None
```


MONITOR
V04-000

- VAX/VMS Performance Monitor Utility^{1 5}
SUMMARY_TOP - Set up Summary for TOP

16-SEP-1984 01:59:24
5-SEP-1984 02:01:24

VAX/VMS Macro V04-00
[MONITOR.SRC]MONITOR.MAR;1

Page 119
(66)

145D 4789 :
145D 4790 :--

MO
VO

```
OFFC 145D 4792 .ENTRY SUMMARY_TOP, *M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      145F 4793
51 00000000'EF D0 145F 4794      MOVL MCAPTR,R1      ; Load MCA pointer
   00 32 A1 07 E5 1466 4795      BBCC #MCASV_TOP_DISP,MCASW_FLAGS(R1),10$ ; Indicate no TOP displays
   00 32 A1 09 E5 146B 4796 10$:      BBCC #MCASV_S_TOP_DISP,MCASW_FLAGS(R1),20$ ; ... done yet for PROCESSES
   00 32 A1 09 E5 146B 4797 20$:      BBCC #MCASV_S_TOP_DISP,MCASW_FLAGS(R1),20$ ; ... or SYSTEM class
   56 04 BC D0 1470 4798      MOVL @4(AP),R6      ; Load CDB pointer
51 00000000'EF D0 1474 4800      MOVL MRBPTR,R1      ; Load MRB pointer
   43 43 A1 00 E1 147B 4801      BBC #MRBSV_DISPLAY,MRBSW_FLAGS(R1),50$ ; No need to clear DATA array
   1480 4802      ; ... if not already used for displaying
52 00000000'EF D0 1480 4803      MOVL SPTR,R2      ; Load SYI pointer
   52 0B A2 3C 1487 4804      MOVZWL MNR,SYSW_MAXPRCCT(R2),R2 ; Get max process count
   58 52 04 C5 148B 4805      MULL3 #4,R2,R8      ; Pass size of DATA array
   10 4B A6 08 E0 148F 4806      BBS #CDBSV_SYSCLS,CDB$L_FLAGS(R6),30$ ; Br if SYSTEM class
   1494 4807
   1494 4808 ;
   1494 4809 ; Call CLEAR_DATA to clear the DATA array for PROCESSES class
   1494 4810 ;
   1494 4811 ;
   59 2E A6 D0 1494 4812      MOVL CDB$A_BUFFERS(R6),R9 ; Pass address of DATA array
   59 08 A9 D0 1498 4813      MOVL MBP$A_DATA(R9),R9 ;
   00000000'EF 16 149C 4814      JSB CLEAR_DATA ; Clear DATA array in preparation for
   14A2 4815 ; ... call to FILL_DISP_BUFF
   14A2 4816 ; (CLEAR_DATA subrtn destroys R0-R5 and
   14A2 4817 ; ... R8,R9)
   1F 11 14A2 4818      BRB 50$ ; Go continue.....
   14A4 4819 ;
   14A4 4820 ;
   14A4 4821 ; Call CLEAR_DATA to clear all 4 DATA arrays for SYSTEM class
   14A4 4822 ;
   14A4 4823 ;
   14A4 4824 30$:
5A 0000003B'EF DE 14A4 4825      MOVAL SYS_TOP_VEC,R10 ; Get addr of vector of ptrs
   5B 58 D0 14AB 4826      MOVL R8,R11 ; Save array length
   57 04 D0 14AE 4827      MOVL #4,R7 ; Number of arrays to clear
   14B1 4828 40$:
   59 6A D0 14B1 4829      MOVL (R10),R9 ; R9 must contain array addr
   58 5B D0 14B4 4830      MOVL R11,R8 ; R8 gets array length
   00000000'EF 16 14B7 4831      JSB CLEAR_DATA ; Clear DATA array in preparation for
   14BD 4832 ; ... call to FDB_SYS_TOP
   14BD 4833 ; (CLEAR_DATA subrtn destroys R0-R5 and
   14BD 4834 ; ... R8,R9)
   5A 10 C0 14BD 4835      ADDL2 #16,R10 ; Point to next array
   EE 57 F5 14C0 4836      SOBGTR R7,40$ ; Loop back to process next one
```



```
14C3 4838 :  
14C3 4839 : Load PROCESSES current collection buffer pointer (MBP$A_BUFFER)  
14C3 4840 : with first collection buffer pointer (MBP$A_BUFFER1ST); then  
14C3 4841 : call FDB_SYS_TOP or FILL_DISP_BUFF to get the 5 TOP arrays  
14C3 4842 : loaded with data from the first collection buffer. Finally,  
14C3 4843 : restore original value of MBP$A_BUFFER for caller's use.  
14C3 4844 :  
14C3 4845 :  
14C3 4846 50$:  
50 00000000'8F DO 14C3 4847 MOVL #<PROCS_CLSNO*CDB$K_SIZE>,R0 ; Compute offset to PROCESSES CDB  
50 00000000'EF40 9E 14CA 4848 MOVAB CDBHEAD[R0],R0 ; Index to CDB address  
52 2E A0 DO 14D2 4849 MOVL CDB$A_BUFFERS(R0),R2 ; Load buffer block ptr  
58 62 DO 14D6 4850 MOVL MBP$A_BUFFER(R2),R8 ; Save current buffer pointer  
62 04 A2 DO 14D9 4851 MOVL MBP$A_BUFFER1ST(R2),MBP$A_BUFFER(R2) ; Move first to current  
1B 4B A6 08 E0 14DD 4852 BBS #CDB$V_SYSCLS,CDB$L_FLAGS(R6),60$ ; Br if SYSTEM class  
14E2 4854  
14E2 4855 ALLOC 4,R0,R1 ; Allocate 12 (4+descr) stack bytes  
14EF 4856 ; ... for FILL_DISP_BUFF call  
61 56 DO 14EF 4857 MOVL R6,(R1) ; CDB pointer into allocated space  
50 DD 14F2 4858 PUSHL R0 ; Push pointer to time quadword  
51 DD 14F4 4859 PUSHL R1 ; Push address of CDB pointer  
F68C CF 02 FB 14F6 4860 CALLS #2,FILL_DISP_BUFF ; Fill 5 TOP arrays  
0D 11 14FB 4861 BRB 70$ ; Go continue.....  
14FD 4862 60$:  
56 DD 14FD 4863 PUSHL R6 ; Pass SYSTEM CDB address  
00000103'EF DF 14FF 4864 PUSHAL FAOSTK ; Pass display buffer address  
FB7F CF 02 FB 1505 4865 CALLS #2,FDB_SYS_TOP ; Fill 4 TOP arrays (doesn't use ORDER)  
150A 4866 70$:  
62 58 DO 150A 4867 MOVL R8,MBP$A_BUFFER(R2) ; Restore current collection buffer ptr  
150D 4868  
50 01 DO 150D 4869 MOVL #SS$ _NORMAL,R0 ; Indicate success  
04 1510 4870 RET ; ... and return
```

```
1511 4872 .SBTTL DISPLAY_PROCS - Put PROCESSES Display Output to Screen
1511 4873
1511 4874 :++
1511 4875
1511 4876 FUNCTIONAL DESCRIPTION:
1511 4877
1511 4878 Issues calls to various SCRPKG routines to display screen
1511 4879 output for the PROCESSES class (regular display).
1511 4880
1511 4881 INPUTS:
1511 4882
1511 4883 4(AP) - address of CDB (Class Descriptor Block) pointer
1511 4884 for the PROCESSES class.
1511 4885
1511 4886 8(AP) - address of quadword containing the system time
1511 4887 value of the latest collection buffer.
1511 4888
1511 4889 IMPLICIT INPUTS:
1511 4890
1511 4891 MCAPTR - pointer to MCA (Monitor Communication Area)
1511 4892 MRBPTR - pointer to MRB (Monitor Request Block)
1511 4893 SPTR - pointer to SYI (System Information Area)
1511 4894
1511 4895 MCASL_PROC_DISP - longword containing number of processes
1511 4896 to display this interval.
1511 4897
1511 4898 PREV_PD - longword containing number of processes
1511 4899 displayed for the previous interval.
1511 4900
1511 4901
1511 4902 OUTPUTS:
1511 4903
1511 4904 None
1511 4905
1511 4906 IMPLICIT OUTPUTS:
1511 4907
1511 4908 Entire display buffer (FAOSTK) displayed to SYS$OUTPUT for
1511 4909 this display event.
1511 4910
1511 4911 PREV_PD -- longword set to the number of processes displayed
1511 4912 this interval, for use next time through.
1511 4913
1511 4914 ROUTINE VALUE:
1511 4915
1511 4916 RO = SS$_NORMAL, or screen package error status.
1511 4917
1511 4918 SIDE EFFECTS:
1511 4919
1511 4920 none
1511 4921
1511 4922 :--
1511 4923
1511 4924 .ENTRY DISPLAY_PROCS, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
1511 4925
1511 4926 MOVL 4(AP),R6 ; Load CDB pointer
1511 4927 MOVL MCAPTR,R11 ; Load MCA pointer
1511 4928 MOVL CDB$A_BUFFERS(R6),R10 ; Load address of buffer block
```

5B 56 04 BC DO 1513 4925
00000000 EF DO 1517 4927
5A 2E A6 DO 151E 4928

OFFC

MONITOR
V04-000

M 5
- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00 Page 123
DISPLAY_PROCS - Put PROCESSES Display On 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 (69)

55	08	AA	D0	1522	4929	MOVL	MBPSA_PR_FAOstk(R10),R5	:	Get pointer to FAO stack
57	18	AB	D0	1526	4930	MOVL	MCASL_PROC_DISP(R11),R7	:	Get number of procs to display
		03	12	152A	4931	BNEQ	10\$	:	Continue if we have some
	0147		31	152C	4932	BRW	110\$	:	Get out quickly if none to display

```
152F 4934 :  
152F 4935 : Compute number of lines to erase (difference in the number of  
152F 4936 : processes from previous display to this one).  
152F 4937 :  
152F 4938 : Clear the display area if necessary.  
152F 4939 :  
152F 4940 :  
152F 4941 10$:  
28 32 AB 59 D4 152F 4942 CLRL R9 : Assume no lines to erase  
06 E1 1531 4943 BBC #MCASV_ERA_SCRL,MCA$W_FLAGS(R11),30$ : If bit clear, no need to erase  
OF 00E4'CF D1 1536 4944 : Previous display a single screen?  
16 15 153B 4945 CMPL W^PREV_PD,#VTOTALINES : Yes -- skip the erase  
01B2 30 154A 4946 BLEQ 20$ : Get 6 bytes for call to BLINK  
OE 50 E8 154D 4947 ALLOC 6,R1,R3 : Erase entire display area  
0135 31 1550 4948 BSBW BLINK : Continue if status OK  
1553 4949 BLBS R0,30$ : Return with status if failed  
1553 4950 BRW DPROCS_RET : (already logged)  
1553 4951 :  
OF 57 D1 1553 4952 20$: CMPL R7,#VTOTALINES : Current display a single screen?  
06 14 1556 4953 BGTR 30$ : No -- continue  
59 00E4'CF 57 C3 1558 4954 SUBL3 R7,W^PREV_PD,R9 : Yes -- calc lines to erase  
155E 4955 :  
155E 4956 :  
155E 4957 : Compute and format ASCII uptime for setup display line.  
155E 4958 :  
155E 4959 :  
155E 4960 :  
155E 4961 30$:  
54 00000000'EF D0 155E 4962 ALLOC 8,R1,R2 : Allocate 8 stack bytes for time calcs  
62 03 A4 7D 156B 4963 MOVL SPTR,R4 : Get SYI pointer  
50 08 BC 7D 1572 4964 MOVQ MNR_SYISQ_BOOTTIME(R4),(R2) : Boot time into calc area  
62 50 C2 1576 4965 MOVQ @8(AP),R0 : Get collection time  
04 A2 51 D9 157A 4966 SUBL2 R0,(R2) : Subtract coll time from boot time  
1581 4967 SBWC R1,4(R2) :  
158E 4968 ALLOC 13,R4,R1 : Get ASCTIM output buffer  
03 50 E8 158E 4969 $ASCTIM_S TIMBUF=(R4), TIMADR=(R2), CVTFLG=#0 : Get ascii uptime  
00E6 31 159D 4970 BLBS R0,40$ : Continue if status OK  
15A0 4971 BRW DPROCS_ERR : Log error & ret with status if failed  
15A3 4972 :  
15A3 4973 :  
15A3 4974 : Put out process count and uptime.  
15A3 4975 :  
15A3 4976 :  
15A3 4977 40$:  
62 57 D0 15A3 4978 MOVL R7,(R2) : Re-use R2 as ptr to $FAOL parm list  
04 A2 54 D0 15A6 4979 MOVL R4,4(R2) : Insert uptime descr ptr into parm list  
15AA 4980 ALLOC 2,R1,R3 : Alloc 2 bytes for DISPLAY_PUT flags  
63 01 B0 15B7 4981 MOVW #1,(R3) : Set bit to force DISPLAY_PUT thru $FAOL  
52 DD 15BA 4982 PUSHL R2 : Push ptr to $FAOL parameter list  
0000261A'EF DD 15BC 4983 PUSHL PROC_SETUP_STR+4 : Push address of setup string  
00002616'EF DF 15C2 4984 PUSHAL PROC_SETUP_STR : Push length of setup string  
53 DD 15C8 4985 PUSHL R3 : Push DISPLAY_PUT request flags  
00001ADA'EF 04 FB 15CA 4986 CALLS #4,DISPLAY_PUT : Put out screen setup string  
03 50 E8 15D1 4987 BLBS R0,50$ : Continue if status OK  
00B1 31 15D4 4988 BRW DPROCS_RET : Return with status if failed (already logg
```



```
15D7 4990 ;
15D7 4991 ; Loop putting out a line for each process
15D7 4992 ;
15D7 4993 ;
15D7 4994 50$:
15D7 4995      ALLOC 8,R1,R2      ; Allocate a descriptor
04 62 0000004B 8F D0 15E4 4996      MOVL #PROC_LINE,(R2)      ; Move const len of FA0 output buff
A2 00001A07'EF D0 15EB 4997      MOVL OUTDSC+4,4(R2)      ; Load ptr to FA0 output buffer
58 57 D0 15F3 4998      MOVL R7,R8      ; Load no. of procs for this interval
OF 58 D1 15F6 4999 60$:
42 15 15F9 5000      CMPL R8,#VTDALINES      ; Will processes fit in display area?
OE 00000000'EF E9 15FB 5001      BLEQ 80$      ; Yes -- go put them there
51 00000000'EF D0 1602 5002      BLBC CTRLCZ_HIT,70$      ; No -- go fill screen if collecting
02 43 A1 05 E0 1602 5003      MOVL MRBPTR,R1      ; Collection has ended
75 11 1609 5004      BBS #MRBSV_DISP_TO_FILE,MRBSW_FLAGS(R1),70$      ; Load MRB pointer
54 0F D0 160E 5005      BRB 130$      ; Go fill screen if output to file
008F 30 1610 5006      ; Quit displaying
70 50 E9 1610 5007      MOVL #VTDALINES,R4      ; Do a screenful
0071 30 1613 5008      BSBW FILL_SCREEN      ;
6A 50 E9 1616 5009      BLBC R0,DPROCS_ERR      ; Exit if error
012A 30 1619 5010      BSBW PRINT_SCREEN      ; Force SCRPKG to display screen
64 50 E9 161C 5011      BLBC R0,DPROCS_ERR      ; Exit if error
012A 30 161F 5012      BSBW HOLD_SCREEN      ; Wait between screenfuls
64 50 E9 1622 5013      BLBC R0,DPROCS_ERR      ; Exit if error
00CA 30 1625 5014      ALLOC 6,R1,R3      ; Get 6 bytes for call to BLINK
50 50 E9 1632 5015      BSBW BLINK      ; Erase entire display area
58 0F C2 1635 5016      BLBC R0,DPROCS_RET      ; Return with status if failed
B9 11 1638 5017      ; (already logged)
SUBL2 #VTDALINES,R8      ; Calculate remaining processes
BRB 60$      ; ... and go get them displayed
```

```

      OE 00000000'EF E9 163D 5027 80$:
      51 00000000'EF D0 163D 5028      BLBC CTRLCZ_HIT,90$      : Go fill screen unless CTRL-C or Z hit
      02 43 A1 05 E0 1644 5029      : Collection has ended
      33 11 1644 5030      MOVL MRBPTR,R1      : Load MRB pointer
      54 58 1648 5031      BBS #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),90$ : Go fill screen if output to file
      2F 50 1650 5032      : Quit displaying
      1650 5033 90$:
      1652 5034      BRB 130$
      54 58 D0 1652 5035      MOVL R8,R4      : Load no. of procs for final screenful
      4E 10 1655 5036      BSBB FILL_SCREEN : ... and put them out
      2F 50 E9 1657 5037      BLBC R0,DPROCS_ERR : Exit if error
      165A 5038
      165A 5039 :
      165A 5040 : At this point, all process lines have been sent to the SCRPKG.
      165A 5041 :
      165A 5042 : If necessary, erase individual screen lines left over from
      165A 5043 : previous display event.
      165A 5044 :
      165A 5045
      53 59 D0 165A 5046      MOVL R9,R3      : Retrieve number of lines to erase
      17 15 165D 5047      BLEQ 110$      : Continue if none to erase
      54 58 08 C1 165F 5048      ADDL3 #FIRST_DATA_LINE,R8,R4 : Compute 1st line to be erased
      1663 5049 100$:
      01 DD 1663 5050      PUSHL #1      : Always erase from column 1
      54 DD 1665 5051      PUSHL R4      : ... at current row
      00000000'GF 02 FB 1667 5052      CALLS #2,G^SCR$ERASE_LINE : Erase current line
      18 50 E9 166E 5053      BLBC R0,DPROCS_ERR : Quit if error
      54 D6 1671 5054      INCL R4      : Get next line number
      ED 53 F5 1673 5055      SOBGTR R3,100$ : Go erase next line
```



```
1676 5057 ;
1676 5058 ; Save number of processes (this interval) for use in next interval.
1676 5059 ;
1676 5060 ;
1676 5061 110$:
00E4'CF 57 D0 1676 5062      MOVL    R7,W^PREV_PD      ; Remember number of procs for next interval
00 32 AB 06 E2 1678 5063      BBSS    #MCASV_ERA_SCRL,MCASW_FLAGS(R11),120$ ; Indicate erase display area next int
1680 5064 ;
1680 5065 120$:
04 08 10 1680 5066      BSBB    PRINT_SCREEN      ; Force SCRPKG to display the screen
04 50 E9 1682 5067      BLBC    R0,DPROCS_ERR      ; Exit if error
1685 5068 130$:
50 01 D0 1685 5069      MOVL    #SS$_NORMAL,R0      ; Successful status
1688 5070 ;
1688 5071 DPROCS_RET:
04 1688 5072      RET
1689 5073 ;
1689 5074 DPROCS_ERR:
03A0 30 1689 5075      BSBW    DISPERR      ; Log display error
04 168C 5076      RET      ; Return with status
168D 5077 ;
168D 5078 ;
168D 5079 ;
168D 5080 ; PRINT_SCREEN subroutine.
168D 5081 ;
168D 5082 ; Forces SCRPKG to actually output entire screen.
168D 5083 ;
168D 5084 ;
168D 5085 PRINT_SCREEN:
168D 5086 ;
00000000'GF 00 FB 168D 5087      CALLS    #0,G^LIB$PUT_BUFFER      ; Output SCRPKG buffer and stop buffering
OD 50 E9 1694 5088      BLBC    R0,10$      ; Exit if error
000020AF'EF 7F 1697 5089      PUSHAQ   SCRDISC      ; Push MONITOR buffer addr
00000000'GF 01 FB 169D 5090      CALLS    #1,G^LIB$SET_BUFFER      ; Set buffering mode again
16A4 5091 10$:
05 16A4 5092      RSB
16A5 5093 ; Return with R0 set
```

```
16A5 5095 :  
16A5 5096 : The FILL_SCREEN subroutine fills the display area with  
16A5 5097 : the number of processes indicated in R4.  
16A5 5098 :  
16A5 5099 : REGISTER INPUTS:  
16A5 5100 :  
16A5 5101 :     R2 -- Pointer to an 8-byte stack area consisting of an FA0  
16A5 5102 :           output descriptor.  
16A5 5103 :           This register not modified by FILL_SCREEN.  
16A5 5104 :  
16A5 5105 :     R4 -- Number of processes to display (less than or equal  
16A5 5106 :           to display area size).  
16A5 5107 :           This register is destroyed by FILL_SCREEN.  
16A5 5108 :  
16A5 5109 :     R5 -- Pointer to current process in display buffer (FA0 Stack).  
16A5 5110 :           On exit, R5 is updated to point to the first process  
16A5 5111 :           to be displayed on the next screen.  
16A5 5112 :  
16A5 5113 : SCRATCH REGISTERS:  
16A5 5114 :  
16A5 5115 :     R1  
16A5 5116 :  
16A5 5117 :     R3 -- Pointer to FA0 control string  
16A5 5118 :  
16A5 5119 :     R10 -- Number of display lines to advance after PUT_LINE.  
16A5 5120 :  
16A5 5121 : RETURN REGISTER:  
16A5 5122 :  
16A5 5123 :     R0 -- On exit, contains status of most recent SCRPKG call.  
16A5 5124 :
```



```
16A5 5126 FILL_SCREEN:
16A5 5127
16A5 5128      PUSHL    #1                ; Set cursor to
01 DD 16A7 5129      PUSHL    #FIRST_DATA_LINE    ; ... first PROCESSES
08 DD 16A9 5130      CALLS    #2,G^SCR$SET_CURSOR ; ... display line
02 FB 16B0 5131      BLBC     R0,40$           ; Exit if error
4B 50 E9 16B3 5132
16B3 5133
5A 01 D0 16B3 5134      MOVL     #1,R10           ; Load number of lines to advance
16B6 5135 10$:
54 01 D1 16B6 5135      CMPL     #1,R4           ; Only one process left to display?
02 12 16B9 5136      BNEQ     20$              ; No -- continue
5A D4 16BB 5137      CLRL      R10              ; Yes -- indicate no advance
16BD 5138
16BD 5139      ; Choose an FAO control string based on process residency
16BD 5140      ;
16BD 5141      ;
16BD 5142      ;
16BD 5143 20$:
53 00002631'EF DE 16BD 5144      MOVAL    PROC_RES_STR,R3      ; Assume this process is resident
3C A5 D5 16C4 5145      TSTL     MNR_PRO$K_FSIZE-4(R5)        ; Is last longword for this process zero?
07 12 16C7 5146      BNEQ     30$              ; No -- process is resident
53 00002676'EF DE 16C9 5147      MOVAL    PROC_NRES_STR,R3     ; Yes -- process is non-resident
16D0 5148      ; Issue FAOL
16D0 5149      ;
16D0 5150      ;
16D0 5151      ;
16D0 5152 30$:
18 50 E9 16D0 5153      $FAOL_S CTRSTR=(R3), OUTBUF=OUTDSC, PRMLST=(R5)
16E3 5154      BLBC     R0,40$           ; Exit if error
16E6 5155
16E6 5156      ; Send process output line to SCRPKG
16E6 5157      ;
16E6 5158      ;
16E6 5159
5A DD 16E6 5160      PUSHL     R10           ; Push number of lines to advance
52 DD 16E8 5161      PUSHL     R2           ; Push addr of FAO output descriptor
02 FB 16EA 5162      CALLS    #2,G^SCR$PUT_LINE    ; Give one process line to SCRPKG
0A 50 E9 16F1 5163      BLBC     R0,40$           ; Exit if error
16F4 5164
55 00000040 8F C0 16F4 5165      ADDL2    #MNR_PRO$K_FSIZE,R5    ; Point to next process in FAO stack
B8 54 F5 16FB 5166      SOBGTR   R4,10$         ; Loop back to do next process
16FE 5167 40$:
05 16FE 5168      RSB                     ; Return
```

```
16FF 5170 :  
16FF 5171 : BLINK Subroutine.  
16FF 5172 :  
16FF 5173 : Erases entire data display area.  
16FF 5174 : Also calls DISPLAY_PUT to replace the status  
16FF 5175 : (footing) line if necessary.  
16FF 5176 :  
16FF 5177 : R3 = Address of 6-byte stack area for call to DISPLAY_PUT:  
16FF 5178 :  
16FF 5179 : 1) longword to hold length of status string;  
16FF 5180 : 2) pair of bytes used for request flags.  
16FF 5181 :  
16FF 5182 : Upon exit, R0 contains status from DISPLAY_PUT call.  
16FF 5183 :  
16FF 5184 : Register R1 is destroyed by this subroutine.  
16FF 5185 :  
16FF 5186 :  
16FF 5187 : BLINK:  
51 00000000'EF 01 DD 16FF 5188 PUSHL #1 ; Column 1  
03 43 A1 08 DD 1701 5189 PUSHL #FIRST_DATA_LINE ; First line of data display area  
6E 02 E1 1703 5190 MOVL MRBPTR,R1 ; Get MRB pointer  
00000000'GF 02 C0 170A 5191 BBC #MRBSV_MFSUM,MRBSW_FLAGS(R1),10$ ; Br if not m.f. summary  
10$: 170F 5192 ADDL2 #2,(SP) ; 1st data line lower for m.f. summary  
1712 5193  
1712 5194 10$: CALLS #2,G^SCR$ERASE_PAGE ; Erase to end of screen  
1719 5195  
1719 5196 :  
1719 5197 : Now replace the footing line that was erased if necessary  
1719 5198 :  
1719 5199 :  
51 50 01 DD 1719 5200 MOVL #SS$ NORMAL,R0 ; Assume normal status  
00000000'EF 01 DD 171C 5201 MOVL MRBPTR,R1 ; Get MRB pointer  
23 43 A1 08 E0 1723 5202 BBS #MRBSV_MFSUM,MRBSW_FLAGS(R1),20$  
43 A1 0E B3 1728 5203 ; Don't refresh status if m.f. summary  
1D 13 1728 5204 BITW #<MRBSM_PLAYBACK+MRBSM_SUMMARY+MRBSM_RECORD>,MRBSW_FLAGS(R1)  
23F7'CF DF 172C 5205 ; Any status fields need refreshing?  
2300'CF 9F 172C 5206 BEQL 20$ ; No -- go exit  
63 22FF'CF 9A 172E 5207 PUSHAL W^STATUS_PARMs ; Push addr of $FAOL parameter list  
53 DD 1732 5208 PUSHAB W^STATUS_STR+1 ; Push address of status string  
04 A3 01 B0 1736 5209 MOVZBL W^STATUS_STR,(R3) ; Load status string length  
00001ADA'EF 04 DF 173B 5210 PUSHL R3 ; Push its address  
04 A3 01 B0 173D 5211 MOVW #1,4(R3) ; Set bit to force DISPLAY_PUT thru $FAOL  
00001ADA'EF 04 DF 1741 5212 PUSHAL 4(R3) ; Push ptr to DISPLAY_PUT request flags  
1744 5213 CALLS #4,DISPLAY_PUT ; Put out status string on bottom line  
174B 5214 20$:  
05 174B 5215 RSB ; Return with status in R0
```



```
174C 5217 :  
174C 5218 : HOLD_SCREEN Subroutine.  
174C 5219 :  
174C 5220 :     Waits after a full screen has been displayed  
174C 5221 :     in order to let the user see it before the  
174C 5222 :     next screenful arrives.  
174C 5223 :  
174C 5224 :     Upon exit, R0 contains status from $SETIMR or $WAITFR.  
174C 5225 :  
174C 5226 :     Register R1 is destroyed by this subroutine.  
174C 5227 :  
174C 5228 :  
174C 5229 : HOLD_SCREEN:  
174C 5230 :  
51 31 50 01 D0 174C 5231      MOVL    #SS$ NORMAL,R0      ; Assume normal status  
    00000000'EF E8 174F 5232      BLBS    CTRLZ_HIT,10$      ; If CTRL-C or Z hit, don't hold  
51 00000000'EF D0 1756 5233      MOVL    MRBPTR,R1      ; Get MRB pointer  
    25 43 A1 05 E0 175D 5234      BBS     #MRB$V_DISP_TO_FILE,MRB$W_FLAGS(R1),10$  
1762 5235      ; Don't hold if output to file  
1762 5236      $SETIMR_S EFN=#BET_EV_FLAG, DAYTIM=VIEWING_DEL  
1777 5237      ; Set time between screens  
    OD 50 E9 1777 5238      BLBC     R0,10$      ; Exit if error  
177A 5239  
177A 5240      $WAITFR_S EFN=#BET_EV_FLAG      ; Wait between screenfuls  
1787 5241 10$:  
05 1787 5242      RSB      ; Return with status
```

```
1788 5244 .SBTTL DISPLAY_TOP - Put PROCESSES/TOP Display Output to Screen
1788 5245
1788 5246 :++
1788 5247 :
1788 5248 : FUNCTIONAL DESCRIPTION:
1788 5249 :
1788 5250 : Constructs FAO control string for a PROCESSES/TOP screen
1788 5251 : and calls DISPLAY_PUT to send the screen to the SCRPKG.
1788 5252 :
1788 5253 : INPUTS:
1788 5254 :
1788 5255 : 4(AP) - address of CDB (Class Descriptor Block) pointer
1788 5256 : for the PROCESSES class.
1788 5257 :
1788 5258 : IMPLICIT INPUTS:
1788 5259 :
1788 5260 : FAOSTK - FAO parameter list for a TOP screen
1788 5261 :
1788 5262 : TOP_PROCS - byte containing count (up to 8) of
1788 5263 : TOP processes to display.
1788 5264 :
1788 5265 : OUTPUTS:
1788 5266 :
1788 5267 : None
1788 5268 :
1788 5269 : IMPLICIT OUTPUTS:
1788 5270 :
1788 5271 : Entire display buffer (FAOSTK) displayed to SYS$OUTPUT for
1788 5272 : this display event.
1788 5273 :
1788 5274 : ROUTINE VALUE:
1788 5275 :
1788 5276 : R0 = SS$_NORMAL, or screen package error status.
1788 5277 :
1788 5278 : SIDE EFFECTS:
1788 5279 :
1788 5280 : none
1788 5281 :
1788 5282 :--
1788 5283 :
1788 5284 :.ENTRY DISPLAY_TOP, ^M<R2,R3,R4,R5,R6,R9,R10,R11>
1788 5285 :
1788 5286 : MOVL @4(AP),R6 ; Load CDB pointer
1788 5287 : MOVL CDB$A_FAOCTR(R6),R11 ; Load addr of FAO control string
1788 5288 :
1788 5289 :
1788 5290 : Loop which concatenates as many FAO control string segments
1788 5291 : as needed to build the portion of the control string for
1788 5292 : processes to be displayed. The portion for null lines is
1788 5293 : built later.
1788 5294 :
1788 5295 :
1788 5296 : MOVB #FIRST_DATA_LINE,W^TOPLNNO ; Load first TOP display line no
1788 5297 : MOVZBL W^TOPSTR,R9 ; Load length of FAO ctr str for 1 line
1788 5298 : MOVZBL W^TOP_PROCS,R10 ; Load number of TOP procs to display
1788 5299 : BEQL 20$ ; Branch if none
1788 5300 10$:
1788 5301
```

56 04 BC D0
5B 04 A6 D0

26C6'CF 08 90
59 26C3'CF 9A
5A 0022'CF 9A
11 13

OE7C


```
6B 26C4'CF 59 28 17A3 5301      MOV C3 R9,W^TOPSTR+1,(R11) ; Move ctr str segment to ctr string
    5B 59 C0 17A9 5302      ADD L2 R9,R11 ; Point to next available byte in ctr str
    26C6'CF 02 80 17AC 5303      ADD B2 #2,W^TOPLNNO ; Update cursor control to next disp line
    EF 5A F5 17B1 5304      SOB GTR R10,10$ ; Move one segment for each process
    17B4 5305
    17B4 5306
    17B4 5307 ; Now loop moving in control string segments for each
    17B4 5308 ; null line to be displayed.
    17B4 5309
    17B4 5310
    17B4 5311 20$:
    17B4 5312      SUB B3 W^TOP_PROCS,CDB$$_ECOUNT(R6),R10 ; Calc number of null lines
    17B8 5313      BE Q L 40$ ; Branch if none
    26F2'CF 26C6'CF 90 17BD 5314      MOV B W^TOPLNNO,W^ERLNNO ; Line number of first null line
    59 26EF'CF 9A 17C4 5315      MOV ZBL W^ERLINE_STR,R9 ; Length of "erase line" control string
    17C9 5316 30$:
    17C9 5317      MOV C3 R9,W^ERLINE_STR+1,(R11) ; Move ctr str segment to ctr string
    5B 59 C0 17CF 5318      ADD L2 R9,R11 ; Update control string pointer
    26F2'CF 02 80 17D2 5319      ADD B2 #2,W^ERLNNO ; Update cursor control to next disp line
    EF 5A F5 17D7 5320      SOB GTR R10,30$ ; Move one segment for each null line
    17DA 5321
    17DA 5322 ; Call DISPLAY_PUT to put the screen image to SYS$OUTPUT.
    17DA 5323
    17DA 5324
    17DA 5325
    17DA 5326 40$:
    17DA 5327      ALLOC 6,R1,R9 ; Alloc 2 bytes for DISPLAY_PUT flags
    17E7 5328 ; ... and a longword for ctr str size
    69 01 90 17E7 5329      MOV B #1,(R9) ; Set bit to force DISPLAY_PUT thru $FAOL
    01 A9 01 90 17EA 5330      MOV B #1,1(R9) ; Force DISPLAY_PUT to output it now
    02 A9 5B 04 A6 C3 17EE 5331      SUB L3 CDB$$_FAOCTR(R6),R11,2(R9) ; Calc size of FAO control string
    17F4 5332
    0103'CF DF 17F4 5333      PUSHAL W^FAOSTK ; Push ptr to $FAOL parameter list
    04 A6 DD 17F8 5334      PUSH L CDB$$_FAOCTR(R6) ; Push address of control string
    02 A9 DF 17FB 5335      PUSHAL 2(R9) ; Push length of control string
    59 DD 17FE 5336      PUSH L R9 ; Push DISPLAY_PUT request flags
    00001ADA'EF 04 FB 1800 5337      CALLS #4,DISPLAY_PUT ; Put out screen setup string
    03 50 E9 1807 5338      BLBC R0,DTOP_RET ; Return with status if failed (already logg
    50 01 D0 180A 5339      MOV L #SS$_NORMAL,R0 ; Indicate success
    180D 5341
    180D 5342 DTOP_RET:
    04 180D 5343      RET ; Return with status set
```

```
180E 5345 .SBTTL DISPLAY_HOMOG - Put Homog Class Display Output to Screen
180E 5346
180E 5347 :++
180E 5348 :
180E 5349 : FUNCTIONAL DESCRIPTION:
180E 5350 :
180E 5351 : Issues calls to various SCRPKG routines to display screen
180E 5352 : output for the current (homogeneous) class.
180E 5353 :
180E 5354 : INPUTS:
180E 5355 :
180E 5356 : 4(AP) - address of CDB (Class Descriptor Block) pointer
180E 5357 : for the current class.
180E 5358 :
180E 5359 : IMPLICIT INPUTS:
180E 5360 :
180E 5361 : MCAPTR - pointer to MCA (Monitor Communication Area)
180E 5362 : MRBPTR - pointer to MRB (Monitor Request Block)
180E 5363 :
180E 5364 : CDX$$_PREV_DCT - longword containing number of elements
180E 5365 : displayed for the previous interval.
180E 5366 :
180E 5367 :
180E 5368 : OUTPUTS:
180E 5369 :
180E 5370 : None
180E 5371 :
180E 5372 : IMPLICIT OUTPUTS:
180E 5373 :
180E 5374 : Entire list of elements for this homogeneous class
180E 5375 : displayed to SYS$OUTPUT for this display event.
180E 5376 :
180E 5377 : CDX$$_PREV_DCT -- longword set to the number of elements displayed
180E 5378 : this interval, for use next time through.
180E 5379 :
180E 5380 : ROUTINE VALUE:
180E 5381 :
180E 5382 : R0 = SS$_NORMAL, or screen package error status.
180E 5383 :
180E 5384 : SIDE EFFECTS:
180E 5385 :
180E 5386 : none
180E 5387 :
180E 5388 : --
180E 5389 :
09E8 180E 5390 .ENTRY DISPLAY_HOMOG, ^M<R3,R5,R6,R7,R8,R11>
1810 5391
5B 56 04 BC D0 1810 5392 MOVL @4(AP),R6 ; Load CDB pointer
57 32 A6 D0 1814 5393 MOVL CDB$$_CDX(R6),R7 ; Load CDX pointer
00000000'EF D0 1818 5394 MOVL MCAPTR,R11 ; Load MCA pointer
181F 5395 ALLOC 6,R1,R3 ; Get 6 bytes for calls to BLINK
182C 5396 ; ... and DISPLAY_PUT
182C 5397
28 A7 D5 182C 5398 TSTL CDX$$_DISPNAM(R7) ; Check if we have a name display rtn
03 12 182F 5399 BNEQ 10$ ; Br if we do
009E 31 1831 5400 BRW 80$ ; Else, go exit if not
```


Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```
1869 5430 :  
1869 5431 : Loop putting out a screenful each time  
1869 5432 :  
1869 5433 :  
1869 5434 35$:  
58 1C A7 D0 1869 5435 MOVL CDXL_DCOUNT(R7),R8 ; Load no. of elements this interval  
55 D4 186D 5436 CLRL R5 ; Init element index  
OF 58 D1 186F 5437 40$:  
39 15 1872 5438 CMPL R8,#VTDATALINES ; Will elements fit on the screen?  
1874 5439 BLEQ 60$ ; Yes -- go put them there  
OE 00000000'EF E9 1874 5440 BLBC CTRLCZ_HIT,50$ ; No -- go fill screen if collecting  
187B 5441 ; Collection has ended  
51 00000000'EF D0 187B 5442 MOVL MRBPTR,R1 ; Load MRB pointer  
02 43 A1 05 E0 1882 5443 BBS #MRBSV_DISP_TO_FILE,MRBSW_FLAGS(R1),50$  
1887 5444 ; Go fill screen if output to file  
53 11 1887 5445 BRB 90$ ; Quit displaying  
1889 5446 50$:  
55 DD 1889 5447 PUSHL R5 ; Push starting element index  
OF DD 188B 5448 PUSHL #VTDATALINES ; ... no. of elts to display  
56 DD 188D 5449 PUSHL R6 ; ... and CDB address  
188F 5450  
00001934'EF 03 FB 188F 5451 CALLS #3,FILL_HOMOG_SCREEN ; Display a screenful of elements  
46 50 E9 1896 5452 BLBC R0,DHOMOG_RET ; Exit if error (already logged)  
55 OF C0 1899 5453 ADDL2 #VTDATALINES,R5 ; Compute element index for next fill  
189C 5454  
FEAD 30 189C 5455 BSBW HOLD_SCREEN ; Wait between screenfuls  
3E 50 E9 189F 5456 BLBC R0,DHOMOG_ERR ; Exit if error  
18A2 5457  
FE5A 30 18A2 5458 BSBW BLINK ; Erase entire display area  
37 50 E9 18A5 5459 BLBC R0,DHOMOG_RET ; Return with status if failed  
18A8 5460 ; (already logged)  
58 OF C2 18A8 5461 SUBL2 #VTDATALINES,R8 ; Calculate remaining elements  
C2 11 18AB 5462 BRB 40$ ; ... and go get them displayed  
18AB 5463
```



```
18AD 5465
18AD 5466 ;
18AD 5467 ; Fill a screen with remaining elements
18AD 5468 ;
18AD 5469
18AD 5470 60$:
OE 00000000'EF E9 18AD 5471 BLBC CTRLCZ_HIT,70$ ; Go fill screen unless CTRL-C or Z hit
18B4 5472 ; Collection has ended
51 00000000'EF D0 18B4 5473 MOVL MRBPTR,R1 ; Load MRB pointer
02 43 A1 05 E0 18BB 5474 BBS #MRBSV_DISP_TO_FILE,MRBSW_FLAGS(R1),70$ ; Go fill screen if output to file
1A 11 18C0 5475 BRB 90$ ; Quit displaying
55 DD 18C2 5476 70$:
58 DD 18C2 5477 PUSH R5 ; Push starting element index
56 DD 18C4 5478 PUSH R8 ; ... no. of elts to display
00001934'EF 03 FB 18C6 5479 PUSH R6 ; ... and CDB address
OD 50 E9 18C8 5480 CALLS #3,FILL_HOMOG_SCREEN ; Display final screenful of elements
18CF 5481 BLBC R0,DHOMOG_RET ; Exit if error (already logged)
18D2 5482
18D2 5483 ;
18D2 5484 ; At this point, all elements have been sent to the SCRPKG.
18D2 5485
18D2 5486
18D2 5487
18D2 5488
```

```
18D2 5490 ;
18D2 5491 ; Save number of elements displayed this interval for use in next interval.
18D2 5492 ;
18D2 5493 ;
18D2 5494 80$:
20 A7 1C A7 D0 18D2 5495      MOVL    CDX$$_DCOUNT(R7), -
00 32 AB 06 E2 18D7 5496      CDX$$_PREV_DCT(R7)      ; Remember no. of elts for next intv'l
18DC 5497      BBSS    #MCAS$_ERA$_SCRL,MCAS$_FLAGS(R11),90$ ; Indicate erase display area next int
18DC 5498
18DC 5499 90$:
50 01 D0 18DC 5500      MOVL    #SS$_NORMAL,R0      ; Successful status
18DF 5501
18DF 5502 DHOMOG_RET:
04 18DF 5503      RET      ; Return with status set
18E0 5504
18E0 5505 DHOMOG_ERR:
0149 30 18E0 5506      BSBW    DISPERR      ; Log display error
04 18E3 5507      RET      ; Return with status
```



```
18E4 5509 :  
18E4 5510 : DISP_HOM_ITMNAM Subroutine.  
18E4 5511 :  
18E4 5512 : Calls DISPLAY_PUT to display the item name string  
18E4 5513 : in the heading for this homogeneous class. It is  
18E4 5514 : entered into the SCRPKG buffer, but not actually  
18E4 5515 : output to the terminal.  
18E4 5516 :  
18E4 5517 : Upon input,  
18E4 5518 :  
18E4 5519 : R1 = Address of 8-byte FAOL parm stack for call to DISPLAY_PUT  
18E4 5520 :  
18E4 5521 : R3 = Address of 6-byte stack area for call to DISPLAY_PUT:  
18E4 5522 :  
18E4 5523 : 1) longword to hold length of status string;  
18E4 5524 : 2) pair of bytes used for request flags.  
18E4 5525 :  
18E4 5526 : R6 = Address of CDB  
18E4 5527 : R7 = Address of CDX  
18E4 5528 :  
18E4 5529 : Upon exit, R0 contains status from DISPLAY_PUT call.  
18E4 5530 :  
18E4 5531 : Register R1 is destroyed by this subroutine.  
18E4 5532 :  
18E4 5533 :  
18E4 5534 : DISP_HOM_ITMNAM:  
18E4 5535 :  
50 2601'CF 06 90 18E4 5536 : MOVB #ILN_REG,W^ITMLNNO ; Set up row no. for reg. displays  
05 00000000'EF D0 18E9 5537 : MOVL MRBPTR,R0 ; Get MRB pointer  
05 43 A0 0B E1 18F0 5538 : BBC #MRB$V_MFSUM,MRB$W_FLAGS(R0),10$ ; Br if not m.f. summary  
2601'CF 02 82 18F5 5539 : SUBB2 #2,W^ITMLNNO ; Adjust row no. for m.f. summary  
18FA 5540 10$:  
61 0000'CF 9A 18FA 5541 : MOVZBL W^NAME_COL,(R1) ; Get col number for name string  
61 61 D7 18FF 5542 : DECL (R1) ; Express as additional spaces  
50 08 A7 9A 1901 5543 : MOVZBL CDX$B_IDISINDEX(R7),R0 ; Get item index for this disp event  
1905 5544 :  
50 1C B640 9A 1905 5545 : MOVZBL @CDB$A_ITMSTR(R6)(R0),R0 ; Load IDB item number  
50 50 11 C4 190A 5546 : MULL2 #IDB$K_ILENGTH,R0 ; Compute index into IDB table  
50 0000'CF40 9E 190D 5547 : MOVAB W^PERFTABLE(R0),R0 ; Address of IDB for this item  
04 A1 04 A0 D0 1913 5548 : MOVL IDB$A_LNAME(R0),4(R1) ; Addr of item name str to FAOL stack  
1918 5549 :  
51 DD 1918 5550 : PUSHL R1 ; Push addr of FAOL parameter list  
191A 5551 :  
63 25FF'CF 9F 191A 5552 : PUSHAB W^ITEM_NAM_STR+1 ; Push addr of item name FAOL ctrl str  
25FE'CF 9A 191E 5553 : MOVZBL W^ITEM_NAM_STR,(R3) ; Load its length  
53 DD 1923 5554 : PUSHL R3 ; Push address of length longword  
04 A3 01 B0 1925 5555 : MOVW #1,4(R3) ; Set bit to force DISPLAY_PUT thru $FAOL  
04 A3 DF 1929 5556 : PUSHAL 4(R3) ; Push ptr to DISPLAY_PUT request flags  
00001ADA'EF 04 FB 192C 5557 : CALLS #4,DISPLAY_PUT ; Put out item name in heading  
1933 5558 :  
05 1933 5559 : RSB ; Return with status in R0
```

```
1934 5561 .SBTTL FILL_HOMOG_SCREEN - Fill a Screen with Homog Class Output
1934 5562 :++
1934 5563 :
1934 5564 : FUNCTIONAL DESCRIPTION:
1934 5565 :
1934 5566 : Issues calls to DISPLAY_PUT to display a full screen
1934 5567 : of output for this homogeneous class.
1934 5568 :
1934 5569 : INPUTS:
1934 5570 :
1934 5571 : 4(AP) - address of CDB (Class Descriptor Block)
1934 5572 : for the current (homogeneous) class.
1934 5573 :
1934 5574 : 8(AP) - number of element names (e.g., disk names) to be displayed.
1934 5575 :
1934 5576 : 12(AP) - Element ID Table index of 1st element to be displayed.
1934 5577 :
1934 5578 : IMPLICIT INPUTS:
1934 5579 :
1934 5580 : MRBPTR - pointer to MRB (Monitor Request Block)
1934 5581 :
1934 5582 : MFSPTR - pointer to MFS (Multi-File Summary Block)
1934 5583 :
1934 5584 : OUTPUTS:
1934 5585 :
1934 5586 : None
1934 5587 :
1934 5588 : IMPLICIT OUTPUTS:
1934 5589 :
1934 5590 : Entire screen full of homogeneous class data, names and
1934 5591 : heading information output to the terminal.
1934 5592 :
1934 5593 : ROUTINE VALUE:
1934 5594 :
1934 5595 : R0 = SS$_NORMAL, or screen package error status.
1934 5596 :
1934 5597 : SIDE EFFECTS:
1934 5598 :
1934 5599 : none
1934 5600 :
1934 5601 : --
1934 5602 :
1934 5603 :
1934 5604 .ENTRY FILL_HOMOG_SCREEN, ^M<R4,R5,R6,R7>
1936 5605
1936 5606 MOVL 4(AP),R6 ; Load CDB pointer
193A 5607 MOVL CDB$A_CDX(R6),R7 ; Load CDX pointer
193E 5608 MOVL 8(AP),R4 ; Load no. of elts to display
1942 5609 BNEQ 10$ ; Br if have some
1944 5610 BRW 70$ ; Else simply go output screen
1947 5611 10$:
1947 5612 MOVL 12(AP),R5 ; Load element index of first one
194B 5613
194B 5614 :
194B 5615 : Set up call to names display routine.
194B 5616 :
194B 5617 :
```

00F0

56	04	AC	DO
57	32	A6	DO
54	08	AC	DO
	03	12	
	00A7	31	
55	0C	AC	DO


```

50 D4 194B 5618 CLRL R0 ; Init count of names to skip displaying
55 D5 194D 5619 TSTL R5 ; First screenful this interval?
0A 12 194F 5620 BNEQ 20$ ; Br if not
OF 20 A7 D1 1951 5621 CMPL CDX$$_PREV_DCT(R7), - ; Previous display a single screen?
1955 5622 #VTDATALINES
50 20 A7 D0 1957 5623 BGTR 20$ ; Br if not
195B 5624 MOVL CDX$$_PREV_DCT(R7),R0 ; Skip display of all "previous" names
54 50 D1 195B 5625 20$: CMPL R0,R4 ; Any additional names this interval?
2A 13 195E 5626 BEQL 40$ ; Skip display if not
7E 08 50 C1 1960 5627 ADDL3 R0,#FIRST_DATA_LINE,-(SP) ; Stack starting row number
51 00000000'EF D0 1964 5628 MOVL MRBPTR,R1 ; Get MRB pointer
03 43 A1 0B E1 196B 5630 BBC #MRB$V_MFSUM,MRB$W_FLAGS(R1),30$ ; Br if not m.f. summary
6E 02 C0 1970 5631 ADDL2 #2,(SP) ; 1st data line lower for m.f. summary
7E 54 50 C3 1973 5632 30$: SUBL3 R0,R4,-(SP) ; Stack name count
7E 55 50 C1 1977 5633 ADDL3 R0,R5,-(SP) ; Stack element index of first ...
56 DD 197B 5635 ; ... name to display
197B 5636 PUSHL R6 ; Stack CDB address
197D 5637
197D 5638 ; Call name display routine
197D 5639
197D 5640
197D 5641
00001A48'EF 04 FB 197D 5642 CALLS #4,DISP_HOM_NAMES ; Display the element names
03 50 E8 1984 5643 BLBS R0,40$ ; Br if OK status
009E 31 1987 5644 BRW FHS_ERR ; Else go exit if error
198A 5645
198A 5646 ; Now display the actual data
198A 5647
198A 5648
198A 5649
198A 5650
198A 5651 ; First, compute the length of the FAO control string
198A 5652
198A 5653
198A 5654 40$: MOVZBL CDB$B_FAOSEGLEN(R6),R1 ; Get length of the FAO segment
51 40 A6 9A 198A 5655 MULL2 R4,R1 ; Compute length of FAO ctrl string
51 54 C4 198E 5656 MOVZBL CDB$B_FAOPRELEN(R6),R0 ; Get length of the FAO prefix
50 41 A6 9A 1991 5657 ADDL3 R0,R1,CDB$$_FAOCTR(R6) ; ... and add it in
66 51 50 C1 1995 5658
1999 5659
1999 5660 ; Alloc some space for DISPLAY_PUT flags
1999 5661
1999 5662
1999 5663
64 01 90 1999 5664 ALLOC 2,R0,R4 ; Alloc 2 bytes for DISPLAY_PUT flags
01 A4 94 19A6 5665 MOVB #1,(R4) ; Set bit to force DISPLAY_PUT thru $FAOL
19A9 5666 CLRB 1(R4) ; ... but don't force to screen yet
19AC 5667
19AC 5668 ; Calculate and stack beginning of FAOSTK segment for this screen
19AC 5669
19AC 5670
19AC 5671
50 00000000'EF D0 19AC 5672 MOVL MRBPTR,R0 ; Get MRB pointer
0E 43 A0 0B E1 19B3 5673 BBC #MRB$V_MFSUM,MRB$W_FLAGS(R0),50$ ; Br if not m.f. summary
50 00000000'EF D0 19B8 5674 MOVL MFSPTR,R0 ; Get MFS pointer
```

```
50 34 A0 04 C5 198F 5675 MULL3 #4,MFSSL_LWORDS(R0),R0 ; Compute bytes in FAOSTK for one elt
OC 11 19C4 5676 BRB 60$ ; Go compute offset
19C6 5677 50$:
00 50 20 D0 19C6 5678 MOVL #<4*TAB_LWORDS>,R0 ; Assume tabular display
42 A6 91 19C9 5679 CMPB CDB$B_ST(R6),#ALL_STAT ; ALL statistic requested?
03 13 19CD 5680 BEQL 60$ ; Br if so
19CF 5681
50 0C D0 19CF 5682 MOVL #<4*BAR_LWORDS>,R0 ; Bar graph display
19D2 5683 60$:
50 55 C4 19D2 5684 MULL2 R5,R0 ; Compute offset to first data to display
7E 00000103'EF40 9E 19D5 5685 MOVAB L^FAOSTK[R0],-(SP) ; ... and stack its address
19DD 5686
04 A6 DD 19DD 5687 PUSHL CDB$A_FAOCTR(R6) ; Stack address of FAO ctrl str
66 DF 19E0 5688 PUSHAL CDB$L_FAOCTR(R6) ; ... and its length
54 DD 19E2 5689 PUSHL R4 ; Stack DISPLAY_PUT request flags
19E4 5690
00001ADA'EF 04 FB 19E4 5691 CALLS #4,DISPLAY_PUT ; Put screenful of homog class data
19EB 5692
39 50 E9 19EB 5693 BLBC R0,FHS_RET ; Return with status if failed
19EE 5694 ; ... (already logged)
19EE 5695
19EE 5696 :
19EE 5697 : Send "REGSET" escape sequence to screen
19EE 5698 : to get back to the regular character set,
19EE 5699 : and force all accumulated output to screen.
19EE 5700 :
19EE 5701
19EE 5702 70$:
19EE 5703 ALLOC 2,R0,R4 ; Alloc 2 bytes for DISPLAY_PUT flags
01 A4 64 94 19FB 5704 CLRB (R4) ; Indicate no $FAOL filter needed
01 90 19FD 5705 MOVB #1,1(R4) ; Force all accumulated output to screen
1A01 5706
1A01 5707 ALLOC 2,R0,R1 ; Get space for esc. seq string & descr
61 0000'8F B0 1A0E 5708 MOVW #REGSET,(R1) ; Move in "reg set" escape sequence
04 A0 DD 1A13 5709 PUSHL 4(R0) ; Stack address of string
60 DF 1A16 5710 PUSHAL (R0) ; ... and its length
54 DD 1A18 5711 PUSHL R4 ; Stack DISPLAY_PUT request flags
1A1A 5712
00001ADA'EF 03 FB 1A1A 5713 CALLS #3,DISPLAY_PUT ; Set reg set and display whole screen
1A21 5714
03 50 E9 1A21 5715 BLBC R0,FHS_RET ; Return with status if failed
1A24 5716 ; ... (already logged)
1A24 5717
50 01 D0 1A24 5718 MOVL #SS$_NORMAL,R0 ; Successful status
1A27 5719
1A27 5720 FHS_RET:
04 1A27 5721 RET ; Return with status set
1A28 5722
1A28 5723 FHS_ERR:
0001 30 1A28 5724 BSBW DISPERR ; Log display error
04 1A2B 5725 RET ; Return with status
```



```
1A2C 5727 :  
1A2C 5728 : DISPERR Subroutine.  
1A2C 5729 :  
1A2C 5730 : Entered when an error has occurred in a system service or  
1A2C 5731 : other routine while attempting to display to the terminal.  
1A2C 5732 : Upon entry, R0 contains the failing status code. This  
1A2C 5733 : routine makes a call to MON_ERR which records a MONITOR  
1A2C 5734 : error code of MNR$_DISPERR and a subordinate error code of  
1A2C 5735 : that in R0. Then, upon exit, the MNR$_DISPERR status is  
1A2C 5736 : placed in R0.  
1A2C 5737 :  
1A2C 5738 : Upon entry,  
1A2C 5739 :  
1A2C 5740 : R0 = Error status code from a system service or other routine  
1A2C 5741 :  
1A2C 5742 : Upon exit,  
1A2C 5743 :  
1A2C 5744 : R0 = MNR$_DISPERR status code  
1A2C 5745 :  
1A2C 5746 : Register R1 is destroyed by this subroutine.  
1A2C 5747 :  
1A2C 5748 :  
1A2C 5749 DISPERR:  
50 DD 1A2C 5750 PUSHL R0 ; Bad status on stack  
6E DF 1A2E 5751 PUSHAL (SP) ; Stack pointer to bad status  
00000000'8F DD 1A30 5752 PUSHL #MNR$_DISPERR ; Stack MONITOR failing status code  
00001EB6'EF 02 FB 1A36 5753 CALLS #2,MON_ERR ; Log the error  
5E 04 C0 1A3D 5754 ADDL2 #4,SP ; Pop original status  
50 00000000'8F D0 1A40 5755 MOVL #MNR$_DISPERR,R0 ; Get new status to caller  
05 1A47 5756 RSB ; Return
```

```
1A48 5758 .SBTTL DISP_HOM_NAMES - Display Names for Homog Class
1A48 5759
1A48 5760 :++
1A48 5761 :
1A48 5762 : FUNCTIONAL DESCRIPTION:
1A48 5763 :
1A48 5764 : Issues calls to SCRPKG routines to display names
1A48 5765 : of homogeneous elements for the current screen.
1A48 5766 : The names are entered into the SCRPKG buffer, but
1A48 5767 : are not actually output to the screen.
1A48 5768 :
1A48 5769 : INPUTS:
1A48 5770 :
1A48 5771 : 4(AP) - address of CDB (Class Descriptor Block)
1A48 5772 : for the current (homogeneous) class.
1A48 5773 :
1A48 5774 : 8(AP) - Element ID Table index of 1st element to be displayed.
1A48 5775 :
1A48 5776 : 12(AP) - number of element names (e.g., disk names) to be displayed.
1A48 5777 :
1A48 5778 : 16(AP) - screen row number on which to display first element.
1A48 5779 :
1A48 5780 : IMPLICIT INPUTS:
1A48 5781 :
1A48 5782 : OUTPUTS:
1A48 5783 :
1A48 5784 : None
1A48 5785 :
1A48 5786 : IMPLICIT OUTPUTS:
1A48 5787 :
1A48 5788 : All names for the current screen full of elements are sent
1A48 5789 : to the SCRPKG.
1A48 5790 :
1A48 5791 : ROUTINE VALUE:
1A48 5792 :
1A48 5793 : R0 = $$$_NORMAL, or screen package error status.
1A48 5794 :
1A48 5795 : SIDE EFFECTS:
1A48 5796 :
1A48 5797 : none
1A48 5798 :
1A48 5799 :--
1A48 5800
07FC 1A48 5801 .ENTRY DISP_HOM_NAMES, *M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
1A4A 5802
56 04 AC D0 1A4A 5803 MOVL 4(AP),R6 ; Load CDB pointer
57 32 A6 D0 1A4E 5804 MOVL CDB$A_CDX(R6),R7 ; Load CDX pointer
59 0C AC D0 1A52 5805 MOVL 12(AP),R9 ; Get number of elements to display
1A56 5806 ALLOC 40,R0,R3 ; Get 10 longwords for an FA0 stack
1A63 5807 ALLOC 10,R0,R2 ; Allocate a descriptor & a word
```



```
1A70 5809 :  
1A70 5810 : Get Element ID Table address of first element to be displayed  
1A70 5811 :  
1A70 5812 :  
5A 58 09 A7 9A 1A70 5813 MOVZBL CDX$B_ELIDLEN(R7),R8 ; Get length of an element ID  
5A 58 08 AC C5 1A74 5814 MULL3 8(AP),R8,R10 ; Compute offset to 1st display elt  
50 51 0C A7 D0 1A79 5815 MOVL CDX$A_ELIDTABLE(R7),R1 ; Get addr of elt ID table  
50 04 00000000'EF D0 1A7D 5816 MOVL MRBPTR,R0 ; Get MRB pointer  
04 43 A0 0B E1 1A84 5817 BBC #MRB$V_MFSUM,MRB$W_FLAGS(R0),10$ ; Br if not m.f. summary  
51 18 A7 D0 1A89 5818 MOVL CDX$A_SELIDTABLE(R7),R1 ; Get addr of super elt ID table  
5A 51 C0 1A8D 5819 10$:  
1A8D 5820 ADDL2 R1,R10 ; Compute addr of 1st display elt  
1A90 5821 :  
1A90 5822 :  
1A90 5823 : Get row and column numbers for first element name  
1A90 5824 :  
1A90 5825 :  
54 10 AC D0 1A90 5826 MOVL 16(AP),R4 ; Get first row number  
55 0000'CF 9A 1A94 5827 MOVZBL W^NAME_COL,R5 ; ... and column number  
1A99 5828 :  
1A99 5829 :  
1A99 5830 : Call class-specific routine to fill the FAO stack  
1A99 5831 : for the current element.  
1A99 5832 :  
1A99 5833 : NOTE -- this routine expects:  
1A99 5834 :  
1A99 5835 : R0,R1 = scratch  
1A99 5836 : R3 = address of 10-longword FAO stack  
1A99 5837 : R6 = address of CDB  
1A99 5838 : R7 = address of CDX  
1A99 5839 : R10 = address of current element ID  
1A99 5840 :  
1A99 5841 :  
28 B7 16 1A99 5842 20$: JSB @CDX$A_DISPNAME(R7) ; Fill the FAO stack  
1A9C 5843 :  
1A9C 5844 :  
1A9C 5845 : FAO stack is set up. Issue the $FAOL and SCR$PUT_SCREEN calls  
1A9C 5846 :  
1A9C 5847 :  
1A9C 5848 :  
1A9C 5849 $FAOL_S CTRSTR=@CDX$A_DISPFAO(R7), OUTLEN=8(R2), -  
1A9C 5850 OUTBUF=W^OUTDSC, PRMLST=(R3) ; Format an element name  
1AAF 5851 :  
04 62 27 50 E9 1AAF 5852 BLBC R0,DHN_RET ; Exit if error  
04 A2 08 A2 3C 1AB2 5853 MOVZWL 8(R2),R2 ; Move actual text len to descr  
1A07'CF D0 1AB6 5854 MOVL W^OUTDSC+4,4(R2) ; Move addr of text to descr  
1ABC 5855 :  
1ABC 5856 :  
1ABC 5857 : Push SCR$PUT_SCREEN arguments on stack and call it to display one name  
1ABC 5858 :  
1ABC 5859 :  
00 DD 1ABC 5860 PUSHL #0 ; No special screen attributes  
55 DD 1ABE 5861 PUSHL R5 ; Column number  
54 DD 1AC0 5862 PUSHL R4 ; Row number  
62 DF 1AC2 5863 PUSHAL (R2) ; Text descriptor  
00000000'GF 04 FB 1AC4 5864 CALLS #4,G^SCR$PUT_SCREEN ; Put a name string to terminal  
1ACB 5865 :
```

OB 50	E9	1ACB	5866	BLBC	R0,DHN_RET	; Exit if error
		1ACE	5867			
SA 58	C0	1ACE	5868	ADDL2	R8,R10	; Point to next element for display
54	D6	1AD1	5869	INCL	R4	; Point to next row number
C3 59	F5	1AD3	5870	SOBGTR	R9,20\$	; ... and go do it
		1AD6	5871			
50 01	D0	1AD6	5872	MOVL	#SS\$,NORMAL,R0	; Successful status
		1AD9	5873			
		1AD9	5874			
	04	1AD9	5875	DHN_RET:		
		1ADA	5876	RET		; Return with status set


```
1ADA 5878      .SBTTL DISPLAY_PUT - Put Display Output to Screen
1ADA 5879
1ADA 5880      :++
1ADA 5881      :
1ADA 5882      : FUNCTIONAL DESCRIPTION:
1ADA 5883      :
1ADA 5884      : Puts DISPLAY output to SYS$OUTPUT (or any file) using
1ADA 5885      : the Screen Package. Depending on the setting of an input
1ADA 5886      : flag, DISPLAY_PUT will either send the supplied buffer
1ADA 5887      : directly on to the screen package, or run it through $FAOL
1ADA 5888      : before sending it. A second input flag indicates whether or
1ADA 5889      : not to actually output the data sent to the screen package.
1ADA 5890
1ADA 5891      : INPUTS:
1ADA 5892      :
1ADA 5893      : 4(AP) - address of 2 contiguous bytes, each containing a flag:
1ADA 5894      :
1ADA 5895      :   Byte 0: If low bit set, use supplied buffer as input
1ADA 5896      :   to $FAOL, and send the resultant buffer
1ADA 5897      :   to the screen package. Otherwise, send
1ADA 5898      :   the buffer directly on to the screen
1ADA 5899      :   package.
1ADA 5900      :
1ADA 5901      :   Byte 1: If low bit set, issue screen package calls to
1ADA 5902      :   actually output the data. Otherwise,
1ADA 5903      :   no such calls are made and the screen
1ADA 5904      :   package merely buffers all received data.
1ADA 5905
1ADA 5906      : 8(AP) - address of longword containing length of buffer to put.
1ADA 5907
1ADA 5908      : 12(AP) - address of buffer to put.
1ADA 5909
1ADA 5910      : 16(AP) - optional address of $FAOL parameter list.
1ADA 5911
1ADA 5912      : IMPLICIT INPUTS:
1ADA 5913      :
1ADA 5914      :   OUTDSC - quadword string descriptor for $FAOL output buffer.
1ADA 5915      :   SCRDSK - quadword string descriptor for buffer required by SCRPKG.
1ADA 5916
1ADA 5917      : OUTPUTS:
1ADA 5918      :
1ADA 5919      :   none
1ADA 5920
1ADA 5921      : IMPLICIT OUTPUTS:
1ADA 5922      :
1ADA 5923      :   Translated buffer sent to Screen Package.
1ADA 5924
1ADA 5925      : ROUTINE VALUE:
1ADA 5926      :
1ADA 5927      :   R0 = SS$_NORMAL, or screen package error status.
1ADA 5928
1ADA 5929      : SIDE EFFECTS:
1ADA 5930      :
1ADA 5931      :   none
1ADA 5932
1ADA 5933      : --
1ADA 5934
```

```
0004 1ADA 5935 .ENTRY DISPLAY_PUT,^M<R2>
      1ADC 5936
      0F 04 BC 00 E0 1ADC 5937 BBS #0,@4(AP),10$ ; Go do $FAOL call if requested
      0C AC DD 1AE1 5938 PUSHL 12(AP) ; Otherwise, simply put buffer
      08 BC DD 1AE4 5939 PUSHL @8(AP) ; ....
00001B5D'EF 02 FB 1AE7 5940 CALLS #2,PUT_TO_SCREEN ; Join common code
      3F 11 1AEE 5941 BRB 20$
      1AF0 5942 10$:
      1AF0 5943 ALLOC 10,R1,R2 ; Allocate a descriptor & a word
      04 62 08 BC D0 1AFD 5944 MOVL @8(AP),(R2) ; Move in length of buffer
      0C AC D0 1B01 5945 MOVL 12(AP),4(R2) ; ... and address of buffer
      3B 50 E9 1B06 5946 $FAOL_S CTRSTR=(R2), OUTLEN=8(R2), OUTBUF=OUTDSC, PRMLST=@16(AP) ;
00001A07'EF DD 1B1B 5947 BLBC R0,DP_ERR ; Exit if error
      7E 08 A2 3C 1B1E 5948 PUSHL OUTDSC+4 ; Push output buffer address
00001B5D'EF 02 FB 1B24 5949 MOVZWL 8(R2),-(SP) ; ... and its length
      27 50 E9 1B28 5950 CALLS #2,PUT_TO_SCREEN ; Put buffer to screen
      7E 04 BC D0 1B2F 5951 20$:
      1A 01 AE 00 E1 1B32 5952 BLBC R0,DP_ERR ; Exit if error
00000000'GF 00 FB 1B36 5953 MOVL @4(AP),-(SP) ; Get parameter bytes on stack
      14 50 E9 1B3B 5954 BBC #0,1(SP),30$ ; Go exit if no output requested
000020AF'EF 7F 1B42 5955 CALLS #0,G^LIB$PUT_BUFFER ; Output SCRPKG buffer and stop buffering
00000000'GF 01 FB 1B45 5956 BLBC R0,DP_ERR ; Exit if error
      04 50 E9 1B48 5957 PUSHAQ SCRDSC ; Push MONITOR buffer addr
      50 01 D0 1B52 5958 CALLS #1,G^LIB$SET_BUFFER ; Set buffering mode again
      04 50 E9 1B55 5959 BLBC R0,DP_ERR ; Exit if error
      50 01 D0 1B58 5960 30$:
      04 1B59 5961 MOVL #SS$_NORMAL,R0 ; No failing status hit
      1B59 5962 RET ; Return with success
      04 1B5C 5963 DP_ERR:
      04 1B5C 5964 BSBW DISPERR ; Log display error
      04 1B5C 5965 RET ; Return with status
```


.SBTTL PUT_TO_SCREEN - Translate escape seqs and issue PUT_SCREEN

1B5D 5967
1B5D 5968
1B5D 5969 :++1B5D 5970 :
1B5D 5971 : FUNCTIONAL DESCRIPTION:1B5D 5972 :
1B5D 5973 : Translate a buffer with imbedded escape sequences to Screen
1B5D 5974 : Package (SCRPKG) calls. The escape sequences present on
1B5D 5975 : input are defined above in the messages declaration section.
1B5D 5976 : They are generally VT52-style sequences, with a few minor
1B5D 5977 : changes. These are converted to general-case SCRPKG calls
1B5D 5978 : to accommodate any terminal. Within the buffer, strings of
1B5D 5979 : text between escape sequences are sent to SCRPKG with a
1B5D 5980 : SCR\$PUT_SCREEN call.1B5D 5981 :
1B5D 5982 : INPUTS:1B5D 5983 :
1B5D 5984 : 4(AP) - length of buffer to translate (longword).
1B5D 5985 : 8(AP) - address of buffer to translate.1B5D 5986 :
1B5D 5987 : IMPLICIT INPUTS:1B5D 5988 :
1B5D 5989 : SYSOUT_TYPE - SYSS\$OUTPUT terminal type (byte).1B5D 5990 :
1B5D 5991 : OUTPUTS:1B5D 5992 :
1B5D 5993 : none1B5D 5994 :
1B5D 5995 : IMPLICIT OUTPUTS:1B5D 5996 :
1B5D 5997 : none1B5D 5998 :
1B5D 5999 : ROUTINE VALUE:1B5D 6000 :
1B5D 6001 : R0 = Worst status received from SCRPKG.1B5D 6002 :
1B5D 6003 : SIDE EFFECTS:1B5D 6004 :
1B5D 6005 : The entire buffer has been sent to the SCRPKG.1B5D 6006 :
1B5D 6007 :--000C 1B5D 6008 :
1B5D 6009 .ENTRY PUT_TO_SCREEN,^M<R2,R3>

52 04 AC 7D 1B5F 6010 MOVQ 4(AP),R2 ; get len & addr of buffer to translate

0000272F'EF 52 7D 1B63 6012 SCANBUF: MOVQ R2,TXT_DESC ; save descriptor of remaining buffer

63 52 1B 3A 1B6A 6014 LOCC #ESC,R2,(R3) ; scan for escape character

52 50 7D 1B6E 6015 MOVQ R0,R2 ; use R2-R3 instead of R0-R1

0000272F'EF 52 A2 1B71 6016 SUBW R2,TXT_LENGTH ; compute length of text

15 13 1B78 6017 BEQL 10\$; br if no text between esc sequences

00000000'GF 0000271B'EF FA 1B7A 6018 CALLG PUTSCRARG,G^SCR\$PUT_SCREEN ; put text string into SCRPKG buffer

07 50 E8 1B85 6019 BLBS R0,10\$; continue if status OK

0000008D'EF 50 D0 1B88 6020 MOVL R0,PTS_STAT ; else, remember it for later

```

52 02 A2 1B8F 6022 10$: SUBW #2,R2 ; update length of remaining buffer
03 18 1B92 6023 BGEQ 20$ ; keep going if more chars in buffer
00DC 31 1B94 6024 BRW PTS_RET ; all done if slid off end
53 02 C0 1B97 6026 20$: ADDL #2,R3 ; update ptr to remaining buffer
2737'CF OA FF A3 3A 1B9A 6027 LOCC -1(R3),#ES_TAB_LEN,W^ESC_SEQ_TABLE ; get offset into tab for CASE
06 12 1BA1 6028 BNEQ 30$ ; go do CASE for known esc sequences
00D5 30 1BA3 6029 BSBW PUT_ESC_SEQ ; unknown esc seq ... just put out as text
00B9 31 1BA6 6031 BRW CHERRET ; join common code
1BA9 6032 30$: CASE R0,<CHEKBUF,PTS_ESCY,PTS_ESCU,PTS_ESCR,PTS_ESCK,-
1BA9 6033 PTS_ESCJ,PTS_ESCH,PTS_ESCG,PTS_ESCF,PTS_ESCB,PTS_ESCL>,W
56 11 1BC3 6034 BRB PTS_ESCH ; if out of range, do a cursor home
1BC5 6035
1BC5 6036 PTS_ESCB: ; set 'bold' attribute
00 272B'CF 00 E2 1BC5 6038 BBSS #SCR$V_BOLD,W^ATTRIBMSK,10$ ; turn on appropriate bit in attrib mask
009E 31 1BCB 6039 10$: BRW CHEKBUF ; join common code
1BCE 6041
1BCE 6042 PTS_ESCL: ; set 'underline' attribute
2615'CF 00 91 1BCE 6043 CMPB #DEC_CRT,W^SYSOUT_TYPE ; is SYS$OUTPUT device VT100-compatible ?
06 12 1BD3 6044 BNEQU 10$ ; no -- don't request underlining
00 272B'CF 03 E2 1BD5 6045 BBSS #SCR$V_UNDERLINE,W^ATTRIBMSK,10$ ; turn on appropriate bit in attrib
008E 31 1BDB 6046 10$: BRW CHEKBUF ; join common code
1BDE 6047
1BDE 6048 PTS_ESCR: ; set 'reverse video' attribute
00 272B'CF 01 E2 1BDE 6050 BBSS #SCR$V_REVERSE,W^ATTRIBMSK,10$ ; turn on appropriate bit in attrib m
0085 31 1BE4 6051 10$: BRW CHEKBUF ; join common code
1BE7 6052
1BE7 6053 PTS_ESCU: ; clear all VT100 attribute settings
272B'CF D4 1BE7 6054 CLRL W^ATTRIBMSK ; do it.....
007E 31 1BEB 6056 BRW CHEKBUF ; join common code
1BEE 6057
1BEE 6058 PTS_ESCY: ; position cursor
52 02 A2 1BEE 6059 SUBW #2,R2 ; update buffer length
03 18 1BF1 6060 BGEQ 10$ ; continue if more buffer left
007D 31 1BF3 6061 BRW PTS_RET ; err if no coordinates -- just quit
1BF6 6062 10$: MOVZBL 1(R3),-(SP) ; stack column number
7E 01 A3 9A 1BF6 6063 MOVZBL (R3),-(SP) ; stack row number
7E 63 9A 1BFA 6064 ADDL #2,R3 ; update ptr to remaining buffer
00000000'GF 53 02 C0 1BFD 6065 CALLS #2,G^SCR$SET_CURSOR ; set the cursor position
02 FB 1C00 6066 BRB CHERRET ; join common code
59 11 1C07 6067
1C09 6068
1C09 6069 PTS_ESCK: ; erase to end of line
00000000'GF 00 FB 1C09 6070 CALLS #0,G^LIB$ERASE_LINE ; do exactly that
50 11 1C10 6071 BRB CHERRET ; join common code
1C12 6072
1C12 6073 PTS_ESCJ: ; erase to end of page (screen)
00000000'GF 00 FB 1C12 6074 CALLS #0,G^LIB$ERASE_PAGE ; do it
47 11 1C19 6075 BRB CHERRET ; join common code
1C1B 6076
1C1B 6077 PTS_ESCH: ; cursor to home
01 DD 1C1B 6078 PUSHL #1 ; stack column number
```



```
00000000'GF 01 DD 1C1D 6079 PUSHL #1 ; stack row number
02 FB 1C1F 6080 CALLS #2,G^SCR$SET_CURSOR ; position cursor to home
3A 11 1C26 6081 BRB CHEKRET ; join common code
1C28 6082
1C28 6083 PTS_ESCF: ; select "alternate" graphics set
0000270D'EF 00002702'EF 7E 1C28 6084 MOVAQ VT100_ALTSET,VT100_CURSET ; make the alternate set current
0B 11 1C33 6085 BRB SELECT_SET ; ... and go output esc seq to select it
1C35 6086
1C35 6087 PTS_ESCG: ; select "regular" graphics set
0000270D'EF 000026F6'EF 7E 1C35 6088 MOVAQ VT100_REGSET,VT100_CURSET ; make the regular set current
1C40 6089
1C40 6090 SELECT_SET:
2615'CF 00 91 1C40 6091 CMPB #DEC_CRT,W^SYSOUT_TYPE ; is SYS$OUTPUT device VT100-compatible ?
0F 12 1C45 6092 BNEQU 10$ ; no -- try another type
0000270D'EF DD 1C47 6093 PUSHL VT100_CURSET ; yes -- push addr of esc seq
00000000'GF 01 FB 1C4D 6094 CALLS #1,G^SCR$PUT_SCREEN ; ... and write it
0C 11 1C54 6095 BRB CHEKRET ; join common return from CASE
1C56 6096 10$:
00002615'EF 01 91 1C56 6097 CMPB #VT5X,SYSOUT_TYPE ; is it VT5x series ?
0D 12 1C5D 6098 BNEQU CHEKBUF ; no -- no need to change char set
1C5F 6099
0019 30 1C5F 6100 BSBW PUT_ESC_SEQ ; write out the esc seq just scanned
1C62 6101
1C62 6102 CHEKRET:
0000008D'EF 07 50 E8 1C62 6103 BLBS R0,CHEKBUF ; continue if status OK
50 D0 1C65 6104 MOVL R0,PTS_STAT ; else, remember it for later
1C6C 6105
1C6C 6106 CHEKBUF: ; common return point for CASE
52 B5 1C6C 6107 TSTW R2 ; whole buffer examined yet ?
03 13 1C6E 6108 BEQL PTS_RET ; yes -- get out
FEF0 31 1C70 6109 BRW SCANBUF ; no -- go look at more
1C73 6110
1C73 6111 PTS_RET:
50 0000008D'EF D0 1C73 6112 MOVL PTS_STAT,R0 ; return status value
04 1C7A 6113 RET
1C7B 6114
1C7B 6115 PUT_ESC_SEQ: ; subroutine to put an imbedded esc
1C7B 6116 ; ... sequence directly to the screen
1C7B 6117
1C7B 6118
0000272F'EF 02 B0 1C7B 6119 MOVW #2,TXT_LENGTH ; load length of esc sequence
00002733'EF FE A3 9E 1C82 6120 MOVAB -2(R3),TXT_START ; load starting address
0000272F'EF 7F 1C8A 6121 PUSHAQ TXT_DESC ; push descriptor address
00000000'GF 01 FB 1C90 6122 CALLS #1,G^SCR$PUT_SCREEN ; ... and put out "as is"
05 1C97 6123 RSB
```

```
1C98 6125      .SBTTL SELECT_REV_LEVS - Select Revision Levels
1C98 6126
1C98 6127      :++
1C98 6128
1C98 6129      : FUNCTIONAL DESCRIPTION:
1C98 6130
1C98 6131
1C98 6132      : This routine is called from the REQUEST_INIT routine in
1C98 6133      : REQUEST.PLI to select the appropriate Revision Level for
1C98 6134      : each class being monitored. Once the level is selected,
1C98 6135      : this routine stores the level number in REVLEVELS, a
1C98 6136      : 128-byte vector which contains level numbers for all
1C98 6137      : classes being monitored. Then it moves the CChange
1C98 6138      : Descriptor (CHD) for each class into its Class Descriptor
1C98 6139      : Block (CDB/CDX). For playback, a class which has a revision
1C98 6140      : level unknown to this version of MONITOR is flagged in
1C98 6141      : the UNK_CLASSES vector.
1C98 6142
1C98 6143      : INPUTS:
1C98 6144
1C98 6145      : 4(AP) - address of a 128-bit vector describing classes
1C98 6146      : to monitor. If class n is to be monitored,
1C98 6147      : bit n is a 1; otherwise it is 0.
1C98 6148
1C98 6149      : 8(AP) - address of a 128-bit vector which will describe
1C98 6150      : classes with revision levels unknown to this
1C98 6151      : version of MONITOR (UNK_CLASSES). It is used only
1C98 6152      : for playback requests. For live requests and playback
1C98 6153      : of Version 3 files (all classes at Rev 0), 8(AP)
1C98 6154      : contains 0. Upon entry, all bits are
1C98 6155      : indeterminate. For each class n to be monitored,
1C98 6156      : bit n is set to 0 if its revision level is known
1C98 6157      : and 1 if its revision level is unknown.
1C98 6158
1C98 6159      : 12(AP) - address of HDR$T_REVLEVELS, a 128-byte vector
1C98 6160      : indicating the revision level of each recorded
1C98 6161      : (i.e., input) class (for playback requests only).
1C98 6162      : For live requests and playback of Version 3 files
1C98 6163      : (all classes at Rev 0), 12(AP) contains 0.
1C98 6164
1C98 6165      : 16(AP) - address of REVLEVELS, a 128-byte vector, into which
1C98 6166      : will be stored a level number for each class being
1C98 6167      : monitored. Upon input, all bytes contain 0.
1C98 6168
1C98 6169      : IMPLICIT INPUTS:
1C98 6170
1C98 6171      : MAX_CLASS_NO - maximum class number defined.
1C98 6172      : MRBPTR - pointer to MRB (Monitor Request Block)
1C98 6173      : CDBHEAD - table of contiguous CDBs.
1C98 6174
1C98 6175      : OUTPUTS:
1C98 6176
1C98 6177      : For each class to be monitored, one of two things happens:
1C98 6178
1C98 6179      : 1) if its revision level is unknown, the appropriate
1C98 6180      : bit in UNK_CLASSES is set. (Can happen only
1C98 6181      : on playback); or,
```



```
1C98 6182 :
1C98 6183 :
1C98 6184 :
1C98 6185 :
1C98 6186 :
1C98 6187 :
1C98 6188 :
1C98 6189 :
1C98 6190 :
1C98 6191 :
1C98 6192 :
1C98 6193 :
1C98 6194 :
1C98 6195 :
1C98 6196 :
1C98 6197 :
1C98 6198 :
1C98 6199 :
1C98 6200 :
1C98 6201 :
1C98 6202 :
1C9A 6203 :
1C9A 6204 :
1CA1 6205 :
1CA4 6206 :
1CA6 6207 :
1CAE 6208 :
1CAE 6209 :
1CAE 6210 :
1CAE 6211 :
1CAE 6212 :
1CAE 6213 :
1CB0 6214 :
1CB0 6215 :
1CB3 6216 :
1CB3 6217 :
1CB6 6218 :
1CB6 6219 :
1CBC 6220 :
1CBC 6221 :
1CBE 6222 :
1CC0 6223 :
1CC3 6224 :
1CC7 6225 :
1CCA 6226 :
1CCC 6227 :
1CCC 6228 :
1CCD 6229 :
1CD4 6230 :
1CD4 6231 :
1CDB 6232 :
1CDC 6233 :
1CDC 6234 :
1CDC 6235 :
1CDC 6236 :
1CDC 6237 :
1CDC 6238 :

2) the CHD (CHange Descriptor) is moved to the CDB/CDX.

Also, for each class to be monitored, the appropriate byte
in REVLEVELS is set to the selected revision level.

IMPLICIT OUTPUTS:
None

ROUTINE VALUE:
NORMAL

SIDE EFFECTS:
None

--
03FC .ENTRY SELECT_REV_LEVS, ^M<R2,R3,R4,R5,R6,R7,R8,R9>
1C9A 6204      MOVL    MRBPTR,R7          ; Get MRB pointer for later use
1CA1 6205      TSTL    8(AP)              ; Check if UNK_CLASSES is provided
1CA4 6206      BEQL    10$                ; If not, don't reference it
1CA6 6207      MOVC5   #0,..#0,#16,a8(AP) ; Assume all classes are NOT unknown
1CAE 6208
1CAE 6209 :
1CAE 6210 : Use FFS instruction to select classes to be monitored.
1CAE 6211 :
1CAE 6212 :
1CAE 6213 10$:
1CAE 6214      CLRL    R5                  ; Init starting bit position
1CB0 6215 20$:
1CB0 6216      MOVL    #32,R3              ; Init bit field size
1CB3 6217      MOVN    R5,R2              ; NOTE -- must handle in 32-bit chunks
1CB6 6218      MOVN    R5,R2              ; Init start position of next chunk
1CB6 6219 30$:
1CB6 6220      FFS     R2,R3,a4(AP),R4     ; Search class bits for next class no.
1CBC 6221      BEQL    40$                ; R4 contains class no. if found
1CBC 6222      BSBB    SELECT_REV         ; Branch if none found this chunk
1CBE 6223      ADDL2   R2,R3              ; Select Rev Level for this class
1CC0 6224      ADDL2   R2,R3              ; Compute next starting
1CC3 6225      ADDL3   #1,R4,R2           ; ... position and field size
1CC7 6226      SUBL2   R2,R3              ; ... for this chunk
1CCA 6227      BRB     30$                ; Go search rest of chunk
1CCC 6228 40$:
1CCC 6229      ACBW    #MAX_CLASS_NO,#32,R5,20$ ; Loop to process next chunk
1CCD 6230
1CD4 6231      MOVL    NORMAL,R0          ; Set normal status
1CDB 6232      RET                                ; Return
1CDC 6233
1CDC 6234
1CDC 6235 SELECT_REV:
1CDC 6236      ; Select Rev Level for this class
1CDC 6237      ; NOTE -- R4 contains class number
1CDC 6238      ; Regs R2, R3, R4, R5 must not be changed
```

```
56 54 00000053 8F C5 1CDC 6239 MULL3 #CDB$K_SIZE,R4,R6 ; Compute offset to desired CDB
56 56 00000000'EF46 9E 1CE4 6240 MOVAB CDBHEAD[R6],R6 ; Index to CDB address
      0B 43 A7 03 E0 1CEC 6241 BBS #MRB$V_PLAYBACK,MRB$W_FLAGS(R7),10$ ; If PLAYBACK, go do it
      58 4F A6 D0 1CF1 6242 ; Else, stay here and do LIVE
      59 88 9A 1CF5 6243 ; Get ptr to CHD header
      4F 10 1CF8 6244 ; Get current rev level
      4C 11 1CFA 6245 ; ... and move CHD for it to CDB/CDX
      1CFC 6246 ; All done with this class
      1CFC 6247 ; Playback
      1CFC 6248 10$:
      58 4F A6 D0 1CFC 6249 ; Get ptr to CHD header
      50 88 9A 1D00 6250 ; Get curr level from CHDHDR
      59 0C AC D0 1D03 6251 20$:
      59 3D 13 1D07 6252 ; Get addr of recorded rev levels
      59 6944 9A 1D09 6253 ; Br if none (recorded lev is 0)
      50 59 D1 1D0D 6254 ; Get recorded rev level
      34 1B 1D10 6255 ; Recorded rev level greater than curr?
      00 08 BC 54 E2 1D12 6256 ; Br if no (and use recorded level)
      1D17 6257 ; Yes -- set bit for this class ...
      1D17 6258 30$:
      28 04 BC 00000000'8F E1 1D17 6259 ; ... in UNK_CLASSES
      00000000'8F 54 D1 1D20 6260 ; Skip checks if no SYSTEM
      12 13 1D27 6261 ; Is this the PROCESSES class?
      00000000'8F 54 D1 1D29 6262 ; Br if yes
      09 13 1D30 6263 ; Is this the STATES class?
      00000000'8F 54 D1 1D32 6264 ; Br if yes
      0D 12 1D39 6265 ; Is this the MODES class?
      1D3B 6266 ; Br if no -- all done with this class
      00 08 BC 00000000'8F E2 1D3B 6267 40$:
      1D44 6268 ; Yes -- also set bit for SYSTEM class
      02 11 1D44 6269 50$:
      1D46 6270 ; ... in UNK_CLASSES
      01 10 1D46 6271 60$:
      1D48 6272 ; All done with this class
      1D48 6273 ; Move CHD for this class to CDB
      1D48 6274 SR_RSB:
      05 1D48 6275 RSB ; Return to caller
      1D49 6276 ;
      1D49 6277 ;
      1D49 6278 ;
      1D49 6279 MOVE_CHD: ; Move CHD for selected rev level ...
      1D49 6280 ; ... to the CDB
      1D49 6281 ;
      1D49 6282 ; Upon input,
      1D49 6283 ;
      1D49 6284 ; R4 = the current class number,
      1D49 6285 ; R6 = addr of CDB for this class.
      1D49 6286 ; R8 = addr of first CHD,
      1D49 6287 ; R9 = the selected Rev Level,
      1D49 6288 ;
      1D49 6289 ; This routine alters R0 and R9.
      1D49 6290 ;
      1D49 6291 ;
      10 BC44 59 90 1D49 6292 MOVAB R9,a16(AP)[R4] ; Set revision level
      59 0D C4 1D4E 6293 MULL2 #CHD$K_SIZE,R9 ; Compute offset to desired CHD
      59 6849 9E 1D51 6294 MOVAB (R8)[R9],R9 ; R9 gets addr of desired CHD
      14 A6 69 D0 1D55 6295 MOVL CHD$L_ICOUNT(R9),CDB$L_ICOUNT(R6) ; Move in item count
```


MONITOR
V04-000

F 8
- VAX/VMS Performance Monitor Utility 16-SEP-1984 01:59:24 VAX/VMS Macro V04-00 Page 155
SELECT_REV_LEVS - Select Revision Levels 5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1 (92)

1C A6	04 A9	D0 1D59	6296	MOVL	CHDSA_ITMSTR(R9),CDBSA_ITMSTR(R6) ; ... and item string ptr
20 A6	08 A9	B0 1D5E	6297	MOVW	CHDSW_BLKLEN(R9),CDBSW_BLKLEN(R6) ; ... and block length
36 A6	0B A9	B0 1D63	6298	MOVW	CHDSW_DISPCTL(R9),CDBSW_DISPCTL(R6) ; ... and display ctl string
09 4B A6	05	E1 1D68	6299	BBC	#CDBSV_HOMOG,CDBSL_FLAGS(R6),10\$; Br if heterogeneous
50	32 A6	D0 1D6D	6300	MOVL	CDBSA_CDX(R6),R0 ; Homogeneous class -- get CDX
09 A0	0A A9	90 1D71	6301	MOVW	CHDSB_ELIDLEN(R9),CDXSB_ELIDLEN(R0) ; Move in elem ID length
		1D76	6302		
		05 1D76	6303	RSB	; Return

```
1D77 6305 .SBTTL ESTAB_CTRL CZ - Establish CTRL-C,Z Handlers
1D77 6306
1D77 6307 :++
1D77 6308 :
1D77 6309 : FUNCTIONAL DESCRIPTION:
1D77 6310 :
1D77 6311 : This routine is called to set up a CTRL-C handler and a
1D77 6312 : CTRL-Z handler for the SYSS$COMMAND terminal device. A
1D77 6313 : channel is assigned to SYSS$COMMAND and its device class
1D77 6314 : is checked for TERMINAL. If not terminal class, the
1D77 6315 : handlers are not established.
1D77 6316 :
1D77 6317 : Then $QIOW's are issued to the terminal driver to establish
1D77 6318 : both handlers. If any system service call fails,
1D77 6319 : the failing status is returned; otherwise, NORMAL status
1D77 6320 : is returned.
1D77 6321 :
1D77 6322 : INPUTS:
1D77 6323 :
1D77 6324 : None
1D77 6325 :
1D77 6326 : IMPLICIT INPUTS:
1D77 6327 :
1D77 6328 : CTRLC - address of CTRL-C handling routine.
1D77 6329 : CTRLZ - address of CTRL-Z handling routine.
1D77 6330 :
1D77 6331 : OUTPUTS:
1D77 6332 :
1D77 6333 : None
1D77 6334 :
1D77 6335 : IMPLICIT OUTPUTS:
1D77 6336 :
1D77 6337 : CTRL-C handler established for "CTRLC" routine.
1D77 6338 : CTRL-Z handler established for "CTRLZ" routine.
1D77 6339 : CTRLCZ_CHAN contains channel number.
1D77 6340 :
1D77 6341 : ROUTINE VALUE:
1D77 6342 :
1D77 6343 : NORMAL, or failing system service status.
1D77 6344 :
1D77 6345 : SIDE EFFECTS:
1D77 6346 :
1D77 6347 : none
1D77 6348 :
1D77 6349 :--
001C 1D77 6350 .ENTRY ESTAB_CTRL CZ, ^M<R2,R3,R4>
1D79 6352
1D79 6353 ALLOC 2,R1,R2 ; Allocate word on stack for chan number
1D86 6354 $ASSIGN,S DEVNAM=W^SYSCMD_DESC, CHAN=(R2) ; Assign channel to SYSS$COMMAND
03 50 E8 1D95 6355 BLBS RO,10$ ; Continue if status OK
008B 31 1D98 6356 BRW EC_ERR ; Branch if error
1D9B 6357
1D9B 6358 10$:
1D9B 6359 ALLOC DIB$K_LENGTH,R3,R4 ; Allocate DIB buffer on stack
1DB0 6360 $GETCHN,S CHAN=(R2), PRIBUF=(R3) ; Get info on SYSS$COMMAND device
61 50 E9 1DC2 6361 BLBC RO,EC_ERR ; Branch if error
```


04 A4	42 8F	91	1DC5	6362			
	53	12	1DC5	6363	CMPB	#DC\$ TERM,DIB\$B_DEVCLASS(R4)	; Is SYS\$COMMAND device a terminal?
			1DCA	6364	BNEQU	EC_NOR	; No -- go return with normal status
000000A1'EF	62	3C	1DCC	6365			
			1DCC	6366	MOVZWL	(R2),L^CTRLCZ_CHAN	; Yes -- save channel no. for \$CANCEL
			1DD3	6367			
			1DD3	6368	\$QIOW_S	CHAN=(R2), -	; Set up CTRL-C handler
			1DD3	6369		FUNC=#<IOS_SETMODE!IOSM_CTRLCAST>, -	
			1DD3	6370		P1=G^CTRLC	
			1DF4	6371			
	2F 50	E9	1DF4	6372	BLBC	RO,EC_ERR	; Branch if error
			1DF7	6373			
			1DF7	6374	\$QIOW_S	CHAN=(R2), -	; Now set up CTRL-Z handler
			1DF7	6375		FUNC=#<IOS_SETMODE!IOSM_OUTBAND>, -	
			1DF7	6376		P1=G^CTRLZ,	
			1DF7	6377		P2=#CTRLZ_MASK	
			1E1C	6378			
	07 50	E9	1E1C	6379	BLBC	RO,EC_ERR	; Branch if error
			1E1F	6380			
50	00000000'EF	D0	1E1F	6381	EC_NOR:	MOVL	NORMAL,RO
			1E1F	6382			; Normal status
			1E26	6383	EC_ERR:		
		04	1E26	6384	RET		; Return with status

```
1E27 6386 .SBTTL ESTAB_CTRLW - Establish CTRL-W Handler
1E27 6387
1E27 6388 :++
1E27 6389
1E27 6390 : FUNCTIONAL DESCRIPTION:
1E27 6391
1E27 6392 : This routine is called to set up a CTRL-W handler for
1E27 6393 : refreshing the display terminal screen. A channel is
1E27 6394 : assigned to the display device and its device class is
1E27 6395 : checked for TERMINAL. If not terminal class, the handler
1E27 6396 : is not established.
1E27 6397
1E27 6398 : Then a $QIOW is issued to the terminal driver to establish
1E27 6399 : the CTRL-W handler. If any system service call fails,
1E27 6400 : the failing status is returned; otherwise, NORMAL status
1E27 6401 : is returned.
1E27 6402
1E27 6403 : INPUTS:
1E27 6404
1E27 6405 : None
1E27 6406
1E27 6407 : IMPLICIT INPUTS:
1E27 6408
1E27 6409 : MRBPTR - pointer to MRB (Monitor Request Block)
1E27 6410
1E27 6411 : CTRLW - address of CTRL-W handling routine.
1E27 6412
1E27 6413 : OUTPUTS:
1E27 6414
1E27 6415 : None
1E27 6416
1E27 6417 : IMPLICIT OUTPUTS:
1E27 6418
1E27 6419 : CTRL-W handler established for "CTRLW" routine.
1E27 6420 : CTRLW_CHAN contains channel number.
1E27 6421
1E27 6422 : ROUTINE VALUE:
1E27 6423
1E27 6424 : NORMAL, or failing system service status.
1E27 6425
1E27 6426 : SIDE EFFECTS:
1E27 6427
1E27 6428 : none
1E27 6429
1E27 6430 :--
1E27 6431
001C 1E27 6432 .ENTRY ESTAB_CTRLW, ^M<R2,R3,R4>
1E29 6433
1E29 6434 ALLOC 2,R1,R2 ; Allocate word on stack for chan number
53 00000000'EF D0 1E36 6435 MOVL MRBPTR,R3 ; Get pointer to MRB
1E3D 6436 $ASSIGN_S DEVNAM=@MRB$A_DISPLAY(R3), - ; Assign channel to display device
1E3D 6437 CHAN=(R2)
67 50 E9 1E4B 6438 BLBC R0,EW_ERR ; Branch if error
1E4E 6439
1E4E 6440 ALLOC DIB$K_LENGTH,R3,R4 ; Allocate DIB buffer on stack
1E63 6441 $GETCHN_S CHAN=(R2), PRIBUF=(R3) ; Get info on display device
3D 50 E9 1E75 6442 BLBC R0,EW_ERR ; Branch if error
```



```
04 A4 42 8F 91 1E78 6443
      2F 12 1E78 6444
      000000A5'EF 62 3C 1E7D 6445
      1E7F 6446
      1E86 6447
      1E86 6448
      1E86 6449
      1E86 6450
      1E86 6451
      1E86 6452
      07 50 E9 1EAB 6453
      1EAB 6454
      1EAE 6455
      50 00000000'EF D0 1EAE 6456 EW_NOR:
      1EAE 6457
      1EB5 6458 EW_ERR:
      04 1EB5 6459 RET
```

```
CMPB #DC$ TERM,DIB$B_DEVCLASS(R4) ; Is display device a terminal?
BNEQU EW_NOR ; No -- go return with normal status

MOVZWL (R2),L^CTRLW_CHAN ; Yes -- save channel no. for $CANCEL

$QIOW_S CHAN=(R2), - ; Set up CTRL-W handler
        FUNC=#<IOS$_SETMODE!IOSM_OUTBAND>, -
        P1=G^CTRLW, -
        P2=#CTRLW_MASK

BLBC R0,EW_ERR ; Branch if error

MOVL NORMAL,R0 ; Normal status

RET ; Return with status
```

```
1EB6 6461 .SBTTL MON_ERR - Log MONITOR Error
1EB6 6462
1EB6 6463 :++
1EB6 6464 :
1EB6 6465 : FUNCTIONAL DESCRIPTION:
1EB6 6466 :
1EB6 6467 : This routine is called to log an error whenever a MONITOR
1EB6 6468 : synchronous error is discovered. Asynchronous (signaled)
1EB6 6469 : errors are logged via the SIGNED_ERR routine.
1EB6 6470 : Logging consists of filling in the PUTMSGVEC array.
1EB6 6471 : This array is the message argument vector for $PUTMSG which
1EB6 6472 : will be called after all routines leading up to this
1EB6 6473 : one have returned.
1EB6 6474 :
1EB6 6475 : INPUTS:
1EB6 6476 :
1EB6 6477 : 4(AP) - MONITOR message code (required parameter)
1EB6 6478 :
1EB6 6479 : 8(AP) - address of a secondary message code (0 if none).
1EB6 6480 : (optional parameter)
1EB6 6481 :
1EB6 6482 : 12(AP) - 1st FAO argument for MONITOR message (optional
1EB6 6483 : parameter). Up to 16 additional FAO arguments
1EB6 6484 : may be included in this parameter list, immediately
1EB6 6485 : following this parameter.
1EB6 6486 :
1EB6 6487 : IMPLICIT INPUTS:
1EB6 6488 :
1EB6 6489 : PUTMSGVEC - 26-longword array, to contain the message argument
1EB6 6490 : vector for $PUTMSG.
1EB6 6491 :
1EB6 6492 : OUTPUTS:
1EB6 6493 :
1EB6 6494 : none
1EB6 6495 :
1EB6 6496 : IMPLICIT OUTPUTS:
1EB6 6497 :
1EB6 6498 : PUTMSGVEC contains message argument vector for later LIB$SIGNAL call.
1EB6 6499 :
1EB6 6500 : ROUTINE VALUE:
1EB6 6501 :
1EB6 6502 : none
1EB6 6503 :
1EB6 6504 : SIDE EFFECTS:
1EB6 6505 :
1EB6 6506 : none
1EB6 6507 :
1EB6 6508 :--
1EB6 6509 :
000C 1EB6 6510 .ENTRY MON_ERR, ^M<R2,R3>
1EB8 6511 :
51 00002745'EF DE 1EB8 6512 MOVAL PUTMSGVEC+4,R1 ; Get pointer to where MONITOR code goes
81 04 AC DO 1EBF 6513 MOVL 4(AP),(R1)+ ; Move it in and point to next item in list
52 6C 3C 1EC3 6514 MOVZWL (AP),R2 ; Get number of input args
52 01 D1 1EC6 6515 CMPL #1,R2 ; Just one arg?
09 19 1EC9 6516 BLSS 10$ ; No -- continue
00002741'EF 01 DO 1ECB 6517 MOVL #1,PUTMSGVEC ; Yes -- tack on argument vector size
```



```

      38  11  1ED2  6518      BRB      ME_RET      ; ... and go return
      1ED4  6519 10$:      SUBL2      #2,R2      ; Compute # of input FAO args
52  02  C2  1ED4  6520      BGTR      20$      ; Continue if found some
      06  14  1ED7  6521      CLRL      (R1)+     ; Indicate none in PUTMSGVEC
      81  D4  1ED9  6522      CLRL      R2       ; Remember for later
      52  D4  1EDB  6523      BRB      50$      ; ... and go check secondary code
      16  11  1EDD  6524      20$:      CMPL      #PUTMSGSIZE-4,R2
      1EDF  6525      BGEQ      30$      ; # FAO args greater than max?
52  16  D1  1EDF  6526      MOVL      #PUTMSGSIZE-4,R2
      03  18  1EE2  6527      ; No, OK as is
52  16  D0  1EE4  6528      30$:      MOVL      R2,(R1)+
      1EE7  6529      ; Move # FAO args into list
      81  52  D0  1EE7  6530      CLRL      R3     ; Clear an index register
      53  D4  1EEA  6531      40$:      MOVL      12(AP)[R3],(R1)+
      1EEC  6532      ; Move an FAO arg into list
81  0C  AC43 D0  1EEC  6533      AOBLS      R2,R3,40$ ; Loop to move all FAO args
F7  53  52  F2  1EF1  6534
      1EF5  6535
      1EF5  6536 50$:      ADDL3      #2,R2,PUTMSGVEC
00002741'EF 52  02  C1  1EF5  6537      TSTL      8(AP) ; Compute # message args and store
      08  AC  D5  1EFD  6538      BEQL      ME_RET ; Secondary message code?
      0A  13  1F00  6539      MOVL      @8(AP),(R1) ; No -- all done
      61  08  BC  D0  1F02  6540      INCL      PUTMSGVEC ; Yes -- move in after FAO args
00002741'EF D6  1F06  6541      ME_RET:      ; ... and count it
      1F0C  6542
      04  1F0C  6543      RET      ; Return to caller
```

```
1F0D 6545 .SBTTL SIGNALLED_ERR - Log Signaled Error
1F0D 6546
1F0D 6547 :++
1F0D 6548 :
1F0D 6549 : FUNCTIONAL DESCRIPTION:
1F0D 6550 :
1F0D 6551 : This routine is called to log an error whenever a MONITOR
1F0D 6552 : asynchronous (signaled) error is discovered. Synchronous
1F0D 6553 : errors (detected by MONITOR) are logged via the MON_ERR
1F0D 6554 : routine. Logging consists of filling in the PUTMSGVEC array.
1F0D 6555 : This array is the message argument vector for $PUTMSG which
1F0D 6556 : will be called after all routines leading up to this
1F0D 6557 : one have returned.
1F0D 6558 :
1F0D 6559 : INPUTS:
1F0D 6560 :
1F0D 6561 : 4(AP) - MONITOR message code (required parameter)
1F0D 6562 :
1F0D 6563 : 8(AP) - secondary message code (required parameter)
1F0D 6564 :
1F0D 6565 : 12(AP) - number of additional (FAO) arguments for secondary
1F0D 6566 : message.
1F0D 6567 :
1F0D 6568 : 16(AP) - address of first additional argument. Others
1F0D 6569 : follow contiguously.
1F0D 6570 :
1F0D 6571 : IMPLICIT INPUTS:
1F0D 6572 :
1F0D 6573 : PUTMSGVEC - 26-longword array, to contain the message argument
1F0D 6574 : vector for $PUTMSG.
1F0D 6575 :
1F0D 6576 : OUTPUTS:
1F0D 6577 :
1F0D 6578 : none
1F0D 6579 :
1F0D 6580 : IMPLICIT OUTPUTS:
1F0D 6581 :
1F0D 6582 : PUTMSGVEC contains message argument vector for later LIB$SIGNAL call.
1F0D 6583 :
1F0D 6584 : ROUTINE VALUE:
1F0D 6585 :
1F0D 6586 : none
1F0D 6587 :
1F0D 6588 : SIDE EFFECTS:
1F0D 6589 :
1F0D 6590 : none
1F0D 6591 :
1F0D 6592 : --
1F0D 6593 :
000C 1F0D 6594 .ENTRY SIGNALLED_ERR, ^M<R2,R3>
1F0F 6595
51 00002745'EF DE 1F0F 6596 MOVAL PUTMSGVEC+4,R1 ; Get pointer to where MONITOR code goes
81 04 AC DO 1F16 6597 MOVL 4(AP),(R1)+ ; Move it in and point to next item in list
81 D4 1F1A 6598 CLRL (R1)+ ; Zero MONITOR FAO args
81 08 AC DO 1F1C 6599 MOVL 8(AP),(R1)+ ; Move in secondary code
00002741'EF 03 DO 1F20 6600 MOVL #3,PUTMSGVEC ; Size of PUTMSGVEC so far
52 D4 1F27 6601 CLRL R2 ; Start out with no PC/PSL args for 2ndary
```



```
00 08 AC 0C 10 ED 1F29 6602 CMPZV #ST$V_FAC_NO,#ST$S_FAC_NO,8(AP),#SYS_FAC_NO ; System fac code?
      03 12 1F2F 6603 BNEQ 10$ ; No -- go add in additional args
      52 02 C0 1F31 6604 ADDL2 #2,R2 ; Yes -- count the PC/PSL args (h'ware xcptn
      10$: 1F34 6605 10$: ADDL2 12(AP),R2 ; Add in caller's additional args for 2ndary
      52 0C AC C0 1F34 6606 BEQL 40$ ; Go exit if none
      19 13 1F38 6607 ; Set up pointer to first add'l arg
      53 10 AC D0 1F3A 6608 MOVL 16(AP),R3 ; # FAO args greater than max?
      52 16 D1 1F3E 6609 CMPL #PUTMSGSIZE-4,R2 ; No, OK as is
      03 18 1F41 6610 BGEQ 20$ ; Yes, replace with max
      52 16 D0 1F43 6611 MOVL #PUTMSGSIZE-4,R2
      20$: 1F46 6612 20$: ADDL2 R2,PUTMSGVEC ; Add the add'l args into PUTMSGVEC size
      00002741'EF 52 C0 1F46 6613 30$: 1F4D 6614 30$: MOVL (R3)+,(R1)+ ; Move from signal array to PUTMSGVEC
      81 83 D0 1F4D 6615 SOBGTR R2,30$ ; Loop once for each add'l arg
      FA 52 F5 1F50 6616 40$: 1F53 6617 40$: RET ; ... and return
      04 1F53 6618
```

```
1F54 6620 .SBTTL SIGNAL_MON_ERR - Signal MONITOR Error
1F54 6621
1F54 6622 :++
1F54 6623 :
1F54 6624 : FUNCTIONAL DESCRIPTION:
1F54 6625 :
1F54 6626 : This routine issues a CALLG to LIB$SIGNAL, passing a
1F54 6627 : signal argument list created by the MON_ERR or SIGNALED_ERR
1F54 6628 : routine. This routine is called from MONMAIN, a PL/I
1F54 6629 : routine; it is necessary because PL/I does not generate
1F54 6630 : G-form routine calls.
1F54 6631 :
1F54 6632 : INPUTS:
1F54 6633 :
1F54 6634 : none
1F54 6635 :
1F54 6636 : IMPLICIT INPUTS:
1F54 6637 :
1F54 6638 : PUTMSGVEC - 26-longword array, containing the signal argument
1F54 6639 : list to be passed to LIB$SIGNAL.
1F54 6640 :
1F54 6641 : OUTPUTS:
1F54 6642 :
1F54 6643 : none
1F54 6644 :
1F54 6645 : IMPLICIT OUTPUTS:
1F54 6646 :
1F54 6647 : Condition is signaled. The VMS default condition handler will
1F54 6648 : display the error messages associated with the condition.
1F54 6649 :
1F54 6650 : ROUTINE VALUE:
1F54 6651 :
1F54 6652 : none
1F54 6653 :
1F54 6654 : SIDE EFFECTS:
1F54 6655 :
1F54 6656 : none
1F54 6657 :
1F54 6658 :--
1F54 6659 :
0000 1F54 6660 .ENTRY SIGNAL_MON_ERR, ^M<>
1F56 6661
00000000'GF 00002741'EF FA 1F56 6662 CALLG L^PUTMSGVEC,G^LIB$SIGNAL ; Signal the MONITOR error
04 1F61 6663 RET ; ... and return
```



```
1F62 6665 .SBTTL LINK_MON_ERR - Link MONITOR Error
1F62 6666
1F62 6667 :++
1F62 6668 :
1F62 6669 : FUNCTIONAL DESCRIPTION:
1F62 6670 :
1F62 6671 : This routine is called to link a MONITOR error message
1F62 6672 : into PUTMSGVEC ahead of the message already there.
1F62 6673 : It uses two input arguments: the MONITOR error message
1F62 6674 : code and the address of its (only) argument.
1F62 6675 :
1F62 6676 : INPUTS:
1F62 6677 :
1F62 6678 : 4(AP) - MONITOR message code
1F62 6679 :
1F62 6680 : 8(AP) - address of the only FA0 argument for the MONITOR
1F62 6681 : message (must be present).
1F62 6682 :
1F62 6683 : IMPLICIT INPUTS:
1F62 6684 :
1F62 6685 : PUTMSGVEC - 26-longword array, to contain the message argument
1F62 6686 : vector for $PUTMSG.
1F62 6687 :
1F62 6688 : OUTPUTS:
1F62 6689 :
1F62 6690 : none
1F62 6691 :
1F62 6692 : IMPLICIT OUTPUTS:
1F62 6693 :
1F62 6694 : PUTMSGVEC updated to include the input MONITOR message as its
1F62 6695 : primary error, followed by the original contents of PUTMSGVEC
1F62 6696 : as a linked error.
1F62 6697 :
1F62 6698 : ROUTINE VALUE:
1F62 6699 :
1F62 6700 : None
1F62 6701 :
1F62 6702 : SIDE EFFECTS:
1F62 6703 :
1F62 6704 : None
1F62 6705 :
1F62 6706 :--
1F62 6707
007C 1F62 6708 .ENTRY LINK_MON_ERR, ^M<R2,R3,R4,R5,R6>
1F64 6709
1F64 6710 ALLOC 4*<PUTMSGSIZE-1>,R0,R6 : Get temp space on stack
1F79 6711 MULL3 #4,PUTMSGVEC,R0 : Compute size of source for move
1F81 6712 MOVCS R0,PUTMSGVEC+4,#0, - : Move current contents to temp area
1F89 6713
1F8D 6714 ADDL2 #4*<PUTMSGSIZE-1>,(R6)
1F8D 6715 #3,PUTMSGVEC : Increase size of vector
1F94 6716 MOVL 4(AP),PUTMSGVEC+4 : Move error code into vector
1F9C 6717 MOVL #1,PUTMSGVEC+8 : Move FA0 arg count into vector
1FA3 6718 MOVL 8(AP),PUTMSGVEC+12 : Move FA0 arg addr into vector
1FAB 6719 MOVCS #4*<PUTMSGSIZE-1>,(R6),#0 - : Move orig contents back in
1FB1 6719 ,#4*<PUTMSGSIZE-4>,PUTMSGVEC+16
1FB9 6720
```

```
50 00002741'EF 04 C5
00 00002745'EF 50 2C
   66 0064 8F
      00002741'EF 03 C0
      00002745'EF 04 AC D0
      00002749'EF 01 D0
      0000274D'EF 08 AC D0
      00 66 0064 8F 2C
00002751'EF 0058 8F
```

Page 166
(98)

			D0	1FB9	6721	\$PUTMSG_S MSGVEC=PUTMSGVEC	; Put out the linked message
	66	01	D0	1FCC	6722	MOVL #1,(R6)	; Use temp area for another msg.
04 A6	00000000	'8F	D0	1FCF	6723	MOVL #MNRS_CONT,4(R6)	; ... which says "Continuing..."
				1FD7	6724		
				1FD7	6725	\$PUTMSG_S MSGVEC=(R6)	; Put it out
			04	1FE6	6726	RET	; Return


```
1FE7 6728 .SBTTL FREE_MEM - Free Virtual Memory
1FE7 6729
1FE7 6730 :++
1FE7 6731 :
1FE7 6732 : FUNCTIONAL DESCRIPTION:
1FE7 6733 :
1FE7 6734 : This routine issues calls to LIB$FREE_VM to free up virtual
1FE7 6735 : memory acquired by classes for FAO control strings and
1FE7 6736 : collection buffer blocks. Also, a special write buffer used
1FE7 6737 : by the PROCESSES class is freed if present; also, the SYSTEM
1FE7 6738 : class DATA arrays. No status code checking is done, since this
1FE7 6739 : routine is in a cleanup path.
1FE7 6740 :
1FE7 6741 : INPUTS:
1FE7 6742 :
1FE7 6743 : None
1FE7 6744 :
1FE7 6745 : IMPLICIT INPUTS:
1FE7 6746 :
1FE7 6747 : The CDB$L_FAOCTR, CDB$A_FAOCTR, CDB$L_BUFFERS and CDB$A_BUFFERS
1FE7 6748 : fields contain the length and address, respectively, of memory
1FE7 6749 : blocks to be freed (for each class in this MONITOR request).
1FE7 6750 :
1FE7 6751 : Additionally, PROC_WRI_BUFD is a quadword containing the length
1FE7 6752 : and address of the special write buffer for the PROCESSES class.
1FE7 6753 :
1FE7 6754 : Also, SYS_DATA_ADDR and SYS_DATA_LEN describe the address and length
1FE7 6755 : of the SYSTEM class DATA arrays.
1FE7 6756 :
1FE7 6757 : OUTPUTS:
1FE7 6758 :
1FE7 6759 : None
1FE7 6760 :
1FE7 6761 : IMPLICIT OUTPUTS:
1FE7 6762 :
1FE7 6763 : Memory is freed. Pointers to freed memory are cleared to 0.
1FE7 6764 :
1FE7 6765 : ROUTINE VALUE:
1FE7 6766 :
1FE7 6767 : NORMAL
1FE7 6768 :
1FE7 6769 : SIDE EFFECTS:
1FE7 6770 :
1FE7 6771 : None
1FE7 6772 :
1FE7 6773 : --
1FE7 6774 :
00FC 1FE7 6775 .ENTRY FREE_MEM, ^M<R2,R3,R4,R5,R6,R7>
1FE9 6776 :
1FE9 6777 : First free up memory left over from a special
1FE9 6778 : write buffer used for recording PROCESSES records.
1FE9 6779 :
1FE9 6780 :
1FE9 6781 :
0000001E'EF D5 1FE9 6782 TSTL L^PROC_WRI_BUFD+4 : Is there a buffer?
19 13 1FE9 6783 BEQL 5$ : Br if not
0000001E'EF DF 1FF1 6784 PUSHAL L^PROC_WRI_BUFD+4 : Yes -- stack addr of buffer ptr
```

```
0000001A'EF DF 1FF7 6785 PUSHAL L^PROC_WRI_BUF D ; Stack addr of buffer length
00000000'GF 02 FB 1FFD 6786 CALLS #2,G^LIB$FREE_VM ; Free the buffer
0000001E'EF D4 2004 6787 CLRL L^PROC_WRI_BUF D+4 ; Clear address
200A 6788
200A 6789 ;
200A 6790 ; Check for SYSTEM class DATA arrays, and free them if present
200A 6791 ;
200A 6792 ;
200A 6793 5$:
0000007B'EF D5 200A 6794 TSTL SYS_DATA_ADDR ; SYSTEM DATA arrays here ?
19 13 2010 6795 BEQL 10$ ; Branch if not
0000007B'EF DF 2012 6796 PUSHAL SYS_DATA_ADDR ; Stack addr of arrays ptr
0000007F'EF DF 2018 6797 PUSHAL SYS_DATA_LEN ; Stack addr of arrays length
00000000'GF 02 FB 201E 6798 CALLS #2,G^LIB$FREE_VM ; Free the space
0000007B'EF D4 2025 6799 CLRL SYS_DATA_ADDR ; Clear address
202B 6800
202B 6801 ;
202B 6802 ; Now look only at the requested classes for this MONITOR request.
202B 6803 ; Free up the FAO control string and the collection buffer block for each.
202B 6804 ;
202B 6805 ;
202B 6806 10$:
57 00000000'EF D0 202B 6808 MOVL MRBPTR,R7 ; Load MRB pointer
55 D4 2032 6809 CLRL R5 ; Init starting bit position
2034 6810 20$:
53 20 D0 2034 6811 MOVL #32,R3 ; Init bit field size
2037 6812 ; NOTE -- must handle in 32-bit chunks
52 55 D0 2037 6813 MOVL R5,R2 ; Init start position of next chunk
203A 6814 30$:
54 32 A7 53 52 EA 203A 6815 FFS R2,R3,MRB$0_CLASSBITS(R7),R4 ; Search for next class number
2040 6816 ; R4 contains class no. if found
OE 13 2040 6817 BEQL 40$ ; Branch if none found this chunk
1C 10 2042 6818 BSBB FREE_CLASS ; Free memory for this class
53 52 C0 2044 6819 ADDL2 R2,R3 ; Compute next starting
52 54 01 C1 2047 6820 ADDL3 #1,R4,R2 ; ... position and field size
53 52 C2 204B 6821 SUBL2 R2,R3 ; ... for this chunk
EA 11 204E 6822 BRB 30$ ; Go search rest of chunk
2050 6823 40$:
FFDC 55 20 0000'8F 3D 2050 6824 ACBW #MAX_CLASS_NO,#32,R5,20$ ; Loop to process next chunk
2058 6825
50 00000000'EF D0 2058 6826 MOVL NORMAL,R0 ; Set normal status
04 205F 6827 RET ; Return
2060 6828
2060 6829
2060 6830 FREE_CLASS: ; Free class memory
2060 6831 ; NOTE -- R4 contains class number
2060 6832
56 54 00000053 8F C5 2060 6833 MULL3 #CDB$K_SIZE,R4,R6 ; Compute offset to desired CDB
56 00000000'EF46 9E 2068 6834 MOVAB CDBHEAD[R6],R6 ; Index to CDB address
04 A6 D5 2070 6835 TSTL CDB$A_FAOCTR(R6) ; Is there an FAO control string?
23 13 2073 6836 BEQL 10$ ; Branch if not
50 00000000'EF DE 2075 6837 MOVAL SYS_FAO_STR,R0 ; Get addr of special SYSTEM FAO str
50 04 A6 D1 207C 6838 CMPL CDB$A_FAOCTR(R6),R0 ; Is this it?
13 13 2080 6839 BEQL 5$ ; Yes, don't try to free it
04 A6 DF 2082 6840 PUSHAL CDB$A_FAOCTR(R6) ; Stack addr of string ptr
66 000005DC 8F D0 2085 6841 MOVL #FAOCTR_SIZE,CDB$L_FAOCTR(R6) ; Ensure whole string is freed
```



```
00000000'GF 66 DF 208C 6842 PUSHAL CDB$$_FAOCTR(R6) ; Stack addr of string length
              02 FB 208E 6843 CALLS #2,G^CIB$FREE_VM ; Free it
              2095 6844 5$:
              04 A6 D4 2095 6845 CLRL CDB$$_FAOCTR(R6) ; Clear address
              2098 6846 10$:
              2E A6 D5 2098 6847 TSTL CDB$$_BUFFERS(R6) ; Is there a buffer block?
              10 13 209B 6848 BEQL 20$ ; Branch if not
              2E A6 DF 209D 6849 PUSHAL CDB$$_BUFFERS(R6) ; Stack addr of block ptr
              2A A6 DF 20A0 6850 PUSHAL CDB$$_BUFFERS(R6) ; Stack addr of block length
00000000'GF 02 FB 20A3 6851 CALLS #2,G^CIB$FREE_VM ; Free it
              2E A6 D4 20AA 6852 CLRL CDB$$_BUFFERSTR6) ; Clear address
              20AD 6853 20$:
              05 20AD 6854 RSB ; Return
```

```
20AE 6856 .SBTTL DISK_DISPNAME - DISK Class display name subroutine
20AE 6857
20AE 6858 :++
20AE 6859 :
20AE 6860 : FUNCTIONAL DESCRIPTION:
20AE 6861 :
20AE 6862 : This subroutine fills an FAO parameter stack with up to
20AE 6863 : 10 longwords required to display a single element (disk)
20AE 6864 : name. The address of an element ID entry is passed to this
20AE 6865 : routine in a register; the address of the FAO stack is
20AE 6866 : also passed in a register.
20AE 6867 :
20AE 6868 : CALLING SEQUENCE:
20AE 6869 :
20AE 6870 : JSB DISK_DISPNAME
20AE 6871 :
20AE 6872 : INPUTS:
20AE 6873 :
20AE 6874 : R3 = address of 10-longword FAO stack
20AE 6875 : R6 = address of CDB
20AE 6876 : R7 = address of CDX
20AE 6877 : R10 = address of current element ID
20AE 6878 :
20AE 6879 : IMPLICIT INPUTS:
20AE 6880 :
20AE 6881 : None
20AE 6882 :
20AE 6883 : OUTPUTS:
20AE 6884 :
20AE 6885 : None
20AE 6886 :
20AE 6887 : IMPLICIT OUTPUTS:
20AE 6888 :
20AE 6889 : The FAO parameter stack is filled with as many longword
20AE 6890 : parameter values as necessary (up to 10) to display a
20AE 6891 : single element name. The number of parameters is defined
20AE 6892 : by the FAO control string defined for this homogeneous
20AE 6893 : class.
20AE 6894 :
20AE 6895 : ROUTINE VALUE:
20AE 6896 :
20AE 6897 : None
20AE 6898 :
20AE 6899 : SIDE EFFECTS:
20AE 6900 :
20AE 6901 : Alters R0.
20AE 6902 :
20AE 6903 :--
```



```
20AE 6905 DISK_DISPNAME::
20AE 6906
20AE 6907
20AE 6908 ; First determine whether we have the special wider name
20AE 6909 ; area available with the tabular (/ALL) statistics display.
20AE 6910
20AE 6911
50 00000000'EF D0 20AE 6912      MOVL      MRBPTR,R0      ; Get MRB pointer
   OA 43 A0 0B E0 20B5 6913      BBS      #MRBSV_MFSUM,MRBSW_FLAGS(R0),10$ ; Br if m.f. summary
   00 42 A6 91 20BA 6914      CMPB      CDB$B_ST(R6),#ALL_STAT ; All statistics requested?
   04 12 20BE 6915      BNEQ      10$      ; Br if no
   50 D4 20C0 6916      CLRL      R0      ; Indicate wide display area
   03 11 20C2 6917      BRB      20$      ; ... and continue
   50 01 D0 20C4 6918 10$:
   32 4B A6 06 E1 20C4 6919      MOVL      #1,R0      ; Indicate narrow display area
   6A 95 20C7 6920 20$:
   2E 13 20C7 6921      BBC      #CDB$V_DISKAC, -      ; Br if recorded without alloc class
   20CC 6922      CDB$L_FLAGS(R6),50$      ;
   20CC 6923      TSTB      (R10)      ; Allocation class 0?
   20CE 6924      BEQL      50$      ; Br if so
   20D0 6925
   20D0 6926 ;
   20D0 6927 ; Process a disk name with allocation class
   20D0 6928 ;
   20D0 6929
2C A7 25BA'CF DE 20D0 6930      MOVAL     W^DISK_FAO_AC, -      ; FAO string with alloc class
   63 6A 9A 20D6 6931      CDX$A_DISPFAO(R7)
   04 A3 01 AA DE 20D6 6932      MOVZBL     (R10),(R3)      ; Move alloc class into FAO stack
   08 A3 05 AA 3C 20D9 6933      MOVAL     1(R10),4(R3)      ; Move in device name pointer
   0C A3 0C A3 D4 20DE 6934      MOVZWL     5(R10),8(R3)      ; ... and unit number
   04 50 E9 20E3 6935
   0C A3 01 D0 20E3 6936      CLRL      12(R3)      ; Assume wide display
   10 A3 D4 20E6 6937      BLBC      R0,30$      ; Br if so
   04 50 E8 20E9 6938      MOVL      #1,12(R3)      ; Else stack a different value
   10 A3 09 D0 20ED 6939 30$:
   14 A3 07 AA 7E 20ED 6940      CLRL      16(R3)      ; Assume zero length node name field
   51 11 20F0 6941      BLBS      R0,40$      ; Br if narrow display area
   20F3 6942      MOVL      #9,16(R3)      ; Stack len of node name field for wide
   20F7 6943 40$:
   20F7 6944      MOVAQ     7(R10),20(R3)      ; Stack node name address
   20FC 6945      BRB      DD_VOL      ; ... and go stack volume name
   20FE 6946
   20FE 6947 ;
   20FE 6948 ; Process a disk name with the node$device format.
   20FE 6949 ;
   20FE 6950
2C A7 2598'CF DE 20FE 6951 50$:
   63 0D D0 20FE 6952      MOVAL     W^DISK_FAO,CDX$A_DISPFAO(R7) ; FAO string without alloc class
   14 A3 01 D0 2104 6953      MOVL      #13,(R3)      ; Assume narrow display area
   06 50 E8 2107 6954      MOVL      #1,20(R3)      ;
   63 16 D0 210B 6955      BLBS      R0,60$      ; Br if narrow
   14 A3 D4 210E 6956      MOVL      #22,(R3)      ; Stack values for wide area
   2111 6957      CLRL      20(R3)      ; .....
   2114 6958 60$:
   1D 4B A6 06 E1 2114 6959      BBC      #CDB$V_DISKAC, -      ; Br if recorded without alloc class
   2119 6960      CDB$L_FLAGS(R6),80$      ; ....
   2119 6961
```

				2119	6962	:	
				2119	6963	:	Process a disk name with with zero allocation class
				2119	6964	:	
				2119	6965	:	
04	A3	07	AA	7E	2119	6966	MOVAQ 7(R10),4(R3) ; Move node name ptr into FA0 stack
		08	A3	D4	211E	6967	CLRL 8(R3) ; Assume zero dollar-sign field length
		07	AA	95	2121	6968	TSTB 7(R10) ; See if node name exists
			04	13	2124	6969	BEQL 70\$; Br if not
	08	A3	01	D0	2126	6970	MOVL #1,8(R3) ; Adjust dollar-sign field length
					212A	6971	70\$:
0C	A3	01	AA	DE	212A	6972	MOVAL 1(R10),12(R3) ; Move in device name pointer
10	A3	05	AA	3C	212F	6973	MOVZWL 5(R10),16(R3) ; ... and unit number
		19		11	2134	6974	BRB DD_VOL ; Go stack volume name


```
2136 6976 :  
2136 6977 : Process a disk name recorded without allocation class.  
2136 6978 : This is the revision level 0 format for the element ID.  
2136 6979 : It does not include an allocation class, and the fields  
2136 6980 : are in positions different from those of later revision  
2136 6981 : levels. This code is also suitable for processing journal  
2136 6982 : device names.  
2136 6983 :  
2136 6984 :  
04 A3 6A 7E 2136 6985 80$:  
      08 A3 D4 2136 6986      MOVAQ (R10),4(R3)      : Move node name ptr into FA0 stack  
      6A 95 213A 6987      CLRL 8(R3)      : Assume zero dollar-sign field length  
      04 13 213D 6988      TSTB (R10)      : See if node name exists  
08 A3 01 D0 213F 6989      BEQL 90$      : Br if not  
      2141 6990      MOVL #1,8(R3)      : Adjust dollar-sign field length  
OC A3 08 AA DE 2145 6991 90$:  
10 A3 0C AA 3C 2145 6992      MOVAL 8(R10),12(R3)      : Move in device name pointer  
      214A 6993      MOVZWL 12(R10),16(R3)      : ... and unit number  
      214F 6994 :  
      214F 6995 DD_VOL:  
1C A3 18 A3 D4 214F 6996      CLRL 24(R3)      : Assume no volume name  
09 4B A6 FE AF D0 2152 6997      MOVL .,28(R3)      : .... (use any accessible address)  
      E1 2157 6998      BBC #CDB$V DISKVN, -      : Br if volume name not available  
      215C 6999      CDB$L FLAGS(R6),DD_RSB      :  
1C A3 18 A3 0C D0 215C 7000      MOVL #12,24(R3)      : Stack length of vol name  
      0F AA 9E 2160 7001      MOVAB 15(R10),28(R3)      : ... and its address  
      2165 7002 :  
      2165 7003 DD_RSB:  
      05 2165 7004      RSB      : Return to caller  
      2166 7005
```

```
2166 7007 .SBTTL SCS_DISPNAME - SCS Class display name subroutine
2166 7008
2166 7009 :++
2166 7010 :
2166 7011 : FUNCTIONAL DESCRIPTION:
2166 7012 :
2166 7013 : This subroutine fills an FAO parameter stack with up to
2166 7014 : 10 longwords required to display a single element (SCS)
2166 7015 : name. The address of an element ID entry is passed to this
2166 7016 : routine in a register; the address of the FAO stack is
2166 7017 : also passed in a register.
2166 7018 :
2166 7019 : CALLING SEQUENCE:
2166 7020 :
2166 7021 : JSB SCS_DISPNAME
2166 7022 :
2166 7023 : INPUTS:
2166 7024 :
2166 7025 : R3 = address of 10-longword FAO stack
2166 7026 : R10 = address of current element ID
2166 7027 :
2166 7028 : IMPLICIT INPUTS:
2166 7029 :
2166 7030 : None
2166 7031 :
2166 7032 : OUTPUTS:
2166 7033 :
2166 7034 : None
2166 7035 :
2166 7036 : IMPLICIT OUTPUTS:
2166 7037 :
2166 7038 : The FAO parameter stack is filled with as many longword
2166 7039 : parameter values as necessary (up to 10) to display a
2166 7040 : single element name. The number of parameters is defined
2166 7041 : by the FAO control string defined for this homogeneous
2166 7042 : class.
2166 7043 :
2166 7044 : ROUTINE VALUE:
2166 7045 :
2166 7046 : None
2166 7047 :
2166 7048 : SIDE EFFECTS:
2166 7049 :
2166 7050 : None
2166 7051 :
2166 7052 :--
2166 7053 :
2166 7054 SCS_DISPNAME::
2166 7055 :
2166 7056 : MOVAQ (R10),(R3) ; Move node name ptr into FAO stack
2166 7057 : TSTB (R10) ; Is there a node name?
2166 7058 : BNEQ 10$ ; Yes, return
2166 7059 : MOVAB UNKNOWN_NODE,(R3) ; No, make it 'Unknown Node'
2174 7060 10$:
2174 7061 : RSB ; Return to caller
2175 7062 :
2175 7063 .END
```

63 6A 7E 2166 7056 MOVAQ (R10),(R3) ; Move node name ptr into FAO stack
6A 95 2166 7057 TSTB (R10) ; Is there a node name?
07 12 2166 7058 BNEQ 10\$; Yes, return
63 000025F1'EF 9E 2166 7059 MOVAB UNKNOWN_NODE,(R3) ; No, make it 'Unknown Node'
05 2174 7060 10\$:
2174 7061 : RSB ; Return to caller
2175 7062 :
2175 7063 .END

MONITOR
Symbol table

M 9
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 175
(103)

```

$ST1          = 00000001
ACT_TO_CUR    = 00000002      G
ADV_HOM_ITEM  = 000006C7      RG      03
ALL_STAT      = 00000000
ALL_TO_ACT    = 00000003      G
ANNCE_STR     = 000022DD      RG      01
ATTRIBMSK     = 0000272B      R       01
AVE_STAT      = 00000002
BART          *****      X       03
BAR10         *****      X       03
BAR11         *****      X       03
BAR2          *****      X       03
BAR3          *****      X       03
BAR4          *****      X       03
BAR5          *****      X       03
BAR6          *****      X       03
BAR7          *****      X       03
BAR8          *****      X       03
BAR9          *****      X       03
BARCHAR       00000005      RG      01
BARHEAD_STR   000024DD      RG      01
BARSIZE       00000001      RG      01
BAR_LWORDS    = 00000003
BET_EV_FLAG   *****      X       03
BET_SCREEN    000000A9      R       01
BLANK_STR     000023F5      RG      01
BLINK         000016FF      R       03
BOT_CURS      000022D6      RG      01
BPU           00000006      R       01
BS_SECS       = 00000002
BU_SYS_SINGLE *****      X       03
CALC_BAR      00000F2E      R       03
CALC_CLASS    0000004B      R       03
CALC_DITEM    000000D2      R       03
CALC_LEN      00000000      RG      03
CB_ADDRS      000000D8      RG      01
CC_ERROR      000003EF      R       03
CC_NORMAL     000003E8      R       03
CDB           = 00000000
CDB$A_BUFFERS = 0000002E
CDB$A_CDX     = 00000032
CDB$A_CHDHDR  = 0000004F
CDB$A_FAOCTR  = 00000004
CDB$A_ITMSTR  = 0000001C
CDB$A_POSTCOLL = 00000026
CDB$A_PRECOLL = 00000022
CDB$A_SUMBUF  = 0000000C
CDB$A_TITLE   = 00000010
CDB$B_FAOPRELEN = 00000041
CDB$B_FAOSEGLEN = 00000040
CDB$B_ST      = 00000042
CDB$B_ST_CUR  = 00000044
CDB$B_ST_DEF  = 00000043
CDB$K_SIZE    = 00000053
CDB$L_BUFFERS = 0000002A
CDB$L_ECOUNTR = 00000018
CDB$L_FAOCTR  = 00000000

```

```

CDB$L_FLAGS   = 0000004B
CDB$L_ICOUNT  = 00000014
CDB$L_MIN     = 00000038
CDB$L_RANGE   = 0000003C
CDB$L_SUMBUF  = 00000008
CDB$M_CPU     = 00000002
CDB$M_CPU_COMB = 00000008
CDB$M_CTPRES  = 00000001
CDB$M_DISABLE = 00000200
CDB$M_DISKAC  = 00000040
CDB$M_DISKVN  = 00000080
CDB$M_EXPLIC  = 00001000
CDB$M_HOMOG   = 00000020
CDB$M_KUNITS  = 00000400
CDB$M_PERCENT = 00000001
CDB$M_STD     = 00000010
CDB$M_SWAPBUF = 00000002
CDB$M_SYSCLS  = 00000100
CDB$M_UNIFORM = 00000004
CDB$M_WIDE    = 00000800
CDB$S_CDB     = 00000053
CDB$S_FILLER  = 00000013
CDB$S_FLAGS   = 00000004
CDB$S_QFILLER = 0000000E
CDB$S_QFLAGS  = 00000002
CDB$V_CPU     = 00000001
CDB$V_CPU_COMB = 00000003
CDB$V_CTPRES  = 00000000
CDB$V_DISABLE = 00000009
CDB$V_DISKAC  = 00000006
CDB$V_DISKVN  = 00000007
CDB$V_EXPLIC  = 0000000C
CDB$V_FILLER  = 0000000D
CDB$V_HOMOG   = 00000005
CDB$V_KUNITS  = 0000000A
CDB$V_PERCENT = 00000000
CDB$V_QFILLER = 00000002
CDB$V_STD     = 00000004
CDB$V_SWAPBUF = 00000001
CDB$V_SYSCLS  = 00000008
CDB$V_UNIFORM = 00000002
CDB$V_WIDE    = 0000000B
CDB$W_BLKLEN  = 00000020
CDB$W_DISPCTL = 00000036
CDB$W_QFLAGS  = 00000045
CDB$W_QFLAGS_CUR = 00000049
CDB$W_QFLAGS_DEF = 00000047
CDBHEAD      *****      X       03
CDBPTR       *****      X       03
CDB_EXT      = 00000000
CDX$A_DISPFAD = 0000002C
CDX$A_DISPNAME = 00000028
CDX$A_ELIDTABLE = 0000000C
CDX$A_ILOOKTAB = 00000024
CDX$A_SCBTABLE = 00000010
CDX$A_SELIDTABLE = 00000018
CDX$B_ELIDLEN = 00000009

```

MONITOR
Symbol table

N 9
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 176
(103)

CDX\$B_IDISCONSEC	= 00000007		
CDX\$B_IDISCT	= 00000006		
CDX\$B_IDISINDEX	= 00000008		
CDX\$K_SIZE	= 00000030		
CDX\$SL_DCOUNT	= 0000001C		
CDX\$SL_PREV_DCT	= 00000020		
CDX\$SL_SELIDTABLE	= 00000014		
CDX\$S_CDB_EXT	= 00000030		
CDX\$S_IBITS	= 00000010		
CDX\$W_CUMELCT	= 0000000A		
CDX\$W_IBITS	= 00000000		
CDX\$W_IBITS_CUR	= 00000004		
CDX\$W_IBITS_DEF	= 00000002		
CHD	= 00000000		
CHD\$A_ITMSTR	= 00000004		
CHD\$B_ELIDLEN	= 0000000A		
CHD\$K_SIZE	= 0000000D		
CHD\$SL_ICOUNT	= 00000000		
CHD\$S_CHD	= 0000000D		
CHD\$W_BLKLEN	= 00000008		
CHD\$W_DISPCTL	= 0000000B		
CHEKBOF	00001C6C	R	03
CHEKRET	00001C62	R	03
CLASS_COLLECT	00000258	RG	03
CLASS_HDR	= 00000000		
CLEAR_DATA	*****	X	03
CLEAR_STACK	00001198	R	03
CLRVTS5	000022B7	RG	01
CLUS\$GL_CLUB	*****	X	03
CLUS_NET_INFO	000005D7	RG	03
CNI_RET	000006C6	R	03
CNI_SUCC	000006BF	R	03
COLLECTION	00000700	R	03
COLLECTION_END	*****	X	03
COLL_BUFS	= 00000002	G	
COLL_COMM	00000764	R	03
COLL_NONSTD	0000078D	R	03
COLL_RSB	0000078C	R	03
COMBINE_MODES	00000423	R	03
COMMON	00000E5C	R	03
COMMON_INIT	00000AA8	R	03
COMM_STR	00002341	RG	01
COMPUTE_BOOTTIME	00000595	RG	03
COMPUTE_STATS	000008DF	R	03
COUNT_TYPE	*****	X	03
CPU_BUSY	*****	X	03
CR	= 0000000D		
CTRLC	*****	X	03
CTRLCZ_CHAN	000000A1	RG	01
CTRLCZ_HIT	*****	X	03
CTRLW	*****	X	03
CTRLW_CHAN	000000A5	RG	01
CTRLW_MASK	00000091	R	01
CTRLZ	*****	X	03
CTRLZ_MASK	00000099	R	01
CURGRAPH	0000000E	RG	01
CURSOR_STR	00002540	RG	01

CURXPOS	00000012	R	01
CUR_STAT	= 00000001		
CUR_TO_ACT	= 00000001	G	
CVT_TO_DELTA	00000587	RG	03
DC\$TERM	= 00000042		
DD_RSB	00002165	R	03
DD_VOL	0000214F	R	03
DEC CRT	= 00000000		
DEF\$A_DISP	= 0000000C		
DEF\$A_REC	= 00000004		
DEF\$A_SUMM	= 00000014		
DEF\$SL_DISP	= 00000008		
DEF\$SL_REC	= 00000000		
DEF\$SL_SUMM	= 00000010		
DEF\$S_DEF_DESC	= 00000018		
DEF_BAR	= 0000002A		
DEF_DESC	= 00000000		
DEF_TO_CUR	= 00000000	G	
DHN_RET	00001AD9	R	03
DHOMOG_ERR	000018E0	R	03
DHOMOG_RET	000018DF	R	03
DIB\$B_DEVCLASS	= 00000004		
DIB\$K_LENGTH	= 00000074		
DISK_DISPNAME	000020AE	RG	03
DISK_FA0	00002598	R	01
DISK_FA0_AC	000025BA	R	01
DISPERR	00001A2C	R	03
DISPLAY_HOMOG	0000180E	RG	03
DISPLAY_INIT	000009A0	RG	03
DISPLAY_PROCS	00001511	RG	03
DISPLAY_PUT	00001ADA	RG	03
DISPLAY_TOP	00001788	RG	03
DISP_HOM_ITMNAME	000018E4	R	03
DISP_HOM_NAMES	00001A48	RG	03
DPROCS_ERR	00001689	R	03
DPROCS_RET	00001688	R	03
DP_ERR	00001B59	R	03
DSC\$K_CLASS_D	= 00000002		
DSC\$K_DTYPE_T	= 0000000E		
DT\$VT100	= 00000060		
DT\$VT52	= 00000040		
DT\$VT55	= 00000041		
DTOP_RET	0000180D	R	03
DYN_STRING	000000FB	RG	01
ECOUNT_SYS_ALL	*****	X	03
ECOUNT_SYS_SINGLE	*****	X	03
EC_ERR	00001E26	R	03
EC_NOR	00001E1F	R	03
ERLINE_STR	000026EF	R	01
ERLNNO	= 000026F2	R	01
ESC	= 0000001B		
ESC_SEQ_TABLE	00002737	R	01
ESTAB_CTRLZ	00001D77	RG	03
ESTAB_CTRLW	00001E27	RG	03
ES_TAB_LEN	= 0000000A		
EW_ERR	00001EB5	R	03
EW_NOR	00001EAE	R	03

MONITOR
Symbol table

B 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 177
(103)

EXESGB_CPUYPE	*****	X	03	IDBSA_ADDR	= 0000000C		
EXESGL_ABSTIM	*****	X	03	IDBSA_LNAME	= 00000004		
EXESGL_RPB	*****	X	03	IDBSA_SNAME	= 00000000		
EXESGQ_SYSTIME	*****	X	03	IDBSB_FLAGS	= 00000010		
FACNO	= 000000CE			IDBSK_ILENGTH	= 00000011		
FAOCTR_SIZE	= 000005DC	G		IDBSM_PCNT	= 00000001		
FAOSTK	00000103	RG	01	IDBSF_FILLER	= 00000007		
FAOSTK_SIZE	= 00001900			IDBSF_FLAGS	= 00000001		
FDB_BEG	00000DBE	R	03	IDBSF_IDB	= 00000011		
FDB_REGPROC	00000C10	R	03	IDBSV_FILLER	= 00000001		
FDB_RET	00000E6B	R	03	IDBSV_PCNT	= 00000000		
FDB_STD	00000CD3	R	03	IDBSW_ISIZE	= 00000008		
FDB_SYS_ALL	00001209	R	03	IDBSW_TYPE	= 0000000A		
FDB_SYS_SINGLE	00000FA4	R	03	ILN_REG	= 00000006		
FDB_SYS_TOP	00001089	RG	03	INTAVE	00000E86	R	03
FETCH	000001DF	RG	03	INTORFL	00000EF3	R	03
FHS_ERR	00001A28	R	03	IOSM_CTRLCAST	= 00000100		
FHS_RET	00001A27	R	03	IOSM_OUTBAND	= 00000400		
FILE_HDR	= 00000000			IOS SETMODE	= 00000023		
FILL_DISP_BUFF	00000B87	RG	03	ITEM_NAM_STR	000025FE	R	01
FILL_HETERO_STATS	000007E9	R	03	ITEM_TYPE	000000E0	R	01
FILL_HOMOG_SCREEN	00001934	RG	03	ITMLNO	= 00002601	R	01
FILL_HOMOG_STATS	*****	X	03	ITMSTR_SYS_ALL	*****	X	03
FILL_PCSTATS_BUFF	000008A8	R	03	ITMSTR_SYS_SINGLE	*****	X	03
FILL_SCREEN	000016A5	R	03	K_STR	0000256E	RG	01
FILL_TOP	000012A7	RG	03	LARGE_NO	= 7FFF7FFF	G	
FIND_TOP	000011A7	R	03	LAST_DATA_LINE	= 00000016	G	
FIN_SEQ	000022C9	RG	01	LF	= 0000000A		
FIRST DATA LINE	= 00000008	G		LIBSERASE_LINE	*****	X	03
FMT_SYS_SINGLE	*****	X	03	LIBSERASE_PAGE	*****	X	03
FOOTP	000023F7	R	01	LIB\$FREE_VM	*****	X	03
FOOTR	000023FF	R	01	LIB\$GET_VM	*****	X	03
FOOTS	000023FB	R	01	LIB\$PUT_BUFFER	*****	X	03
FREE_CLASS	00002060	R	03	LIB\$SET_BUFFER	*****	X	03
FREE-MEM	00001FE7	RG	03	LIB\$SIGNAL	*****	X	03
FSS_BEG	00001056	R	03	LINK_MON_ERR	00001F62	RG	03
FS_ALL	00000FE0	R	03	MAX55HEIGHT	= 000000C8		
FS_AVE	00000FEE	R	03	MAXBARS	= 00000028	G	
FS_COMMON	00001018	R	03	MAXBARS_SYS	= 0000001A	G	
FS_CUR	00000FE0	R	03	MAXELTS	= 000000C8	G	
FS_MAX	0000100A	R	03	MAXELTS_MFS	= 00000190	G	
FS_MIN	00000FFC	R	03	MAX_CLASS_NO	*****	X	03
FT_CASE	000012D3	R	03	MAX_ELIDLEN	= 0000001B	G	
F_ALL	00000DE3	R	03	MAX_HOM_ITEMS	= 0000000F	G	
F_AVE	00000DFF	R	03	MAX_REC_SIZE	= 00007D00	G	
F_CUR	00000DE3	R	03	MAX_STAT	= 00000004		
F_MAX	00000E44	R	03	MBP	= 00000000		
F_MIN	00000E28	R	03	MBP\$A_ADDR	= 00000018		
GET_BUFFERS	*****	X	03	MBP\$A_B1ST	= 00000004		
GET_COMPUTED_ITEMS	0000087D	R	03	MBP\$A_BA	= 00000000		
GMIN	0000000A	R	01	MBP\$A_BUFF1ST	= 00000004		
HARDCOPY	= 00000002			MBP\$A_BUFFERA	= 00000000		
HOLD_SCREEN	0000174C	R	03	MBP\$A_BUFFERA	= 00000000		
HOMOG TYPE	000000E2	R	01	MBP\$A_BUFFERB	= 00000004		
HOM CLASS_PRE	= 00000000			MBP\$A_DATA	= 00000008		
HORIZ_STR	00002542	RG	01	MBP\$A_DIFF	= 0000000C		
IDB	= 00000000			MBP\$A_MAX	= 00000010		

MONITOR
Symbol table

- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 178
(103)

```

MBPSA_MIN      = 0000000C
MBPSA_ORDER    = 00000010
MBPSA_PCMAX    = 00000020
MBPSA_PCMIN    = 0000001C
MBPSA_PCSTATS  = 00000018
MBPSA_PCSUM    = 00000024
MBPSA_PID      = 00000014
MBPSA_PR_FAOSTK = 00000008
MBPSA_STATS    = 00000008
MBPSA_SUM      = 00000014
MBPSK_SIZE     = 00000028
MBPSS_MBP      = 00000028
MBPSS_MBP2     = 0000001C
MBPSS_MBP3     = 0000000C
MBP2           = 00000000
MBP3           = 00000000
MCA            = 00000000
MCASA_INPUT_PTR = 00000004
MCASA_MPADDR   = 0000001C
MCASB_FIRSTC   = 00000030
MCASB_LASTC    = 00000031
MCASK_SIZE     = 0000003A
MCASL_COLLCNT  = 0000000C
MCASL_CONSEC_REC = 00000034
MCASL_DISPCNT  = 00000010
MCASL_INPUT_LEN = 00000000
MCASL_INTTICKS = 00000008
MCASL_INT_MULT = 00000014
MCASL_PROC_DISP = 00000018
MCASQ_CURR_TIME = 00000020
MCASQ_LASTCOLL = 00000028
MCASS_CURR_TIME = 00000008
MCASS_FILLER   = 00000006
MCASS_FLAGS    = 00000002
MCASS_LASTCOLL = 00000008
MCASS_MCA      = 0000003A
MCASV_ENTRY    = 00000000
MCASV_EOF      = 00000003
MCASV_ERA_SCRL = 00000006
MCASV_FILLER   = 0000000A
MCASV_FUTURE   = 00000001
MCASV_GRAPHICS = 00000005
MCASV_MULTFND  = 00000002
MCASV_REFRESH  = 00000008
MCASV_S_TOP_DISP = 00000009
MCASV_TOP_DISP = 00000007
MCASV_VIDEO    = 00000004
MCASW_DCLASSCT = 00000038
MCASW_FLAGS    = 00000032
MCAPTR        = *****
ME_RET         = 00001F0C
MFS            = 00000000
MFSSA_IFB_TAB  = 00000028
MFSSA_STATSBUF = 00000024
MFSSA_SUMMARY  = 0000002C
MFSSB_COLUMNS  = 0000003A
MFSSB_CUR_COL  = 0000003B

```

R X 03
03

```

MFSSB_DATA_COLS = 0000003C
MFSSK_SIZE      = 0000003D
MFSSL_ELEMS     = 00000030
MFSSL_LWORDS    = 00000034
MFSSL_STATSBUF  = 00000020
MFSSQ_CLASSBITS = 00000000
MFSSQ_BEGINNING = 00000010
MFSSQ_ENDING    = 00000018
MFSSS_BEGINNING = 00000008
MFSSS_CLASSBITS = 00000010
MFSSS_ENDING    = 00000008
MFSSS_MFS       = 0000003D
MFSSW_CLASSCT   = 00000038
MFSPTR          = *****
MIN_STAT        = 00000003
MNR$_CONT       = *****
MNR$_DISPERR    = *****
MNR$_ITMNOTDEF  = *****
MNR$_OPENIN     = 00CE109A
MNR$_PREMEOF    = *****
MNR_CLSSB_TYPE  = 00000000
MNR_CLSSK_HSIZE = 0000000D
MNR_CLSSQ_STAMP = 00000003
MNR_CLSSS_CLASS_HDR = 0000000D
MNR_CLSSS_FILLER = 0000000F
MNR_CLSSS_FLAGS = 00000002
MNR_CLSSS_STAMP = 00000008
MNR_CLSSV_CONT  = 00000000
MNR_CLSSV_FILLER = 00000001
MNR_CLSSW_FLAGS = 00000001
MNR_CLSSW_RESERVED = 0000000B
MNR_HDR$B_TYPE  = 00000000
MNR_HDR$K_CLASSBITS = 00000073
MNR_HDR$K_MAXCOMLEN = 0000003C
MNR_HDR$K_REVLEVELS = 00000083
MNR_HDR$K_SIZE  = 00000103
MNR_HDR$L_FLAGS = 00000001
MNR_HDR$L_INTERVAL = 00000015
MNR_HDR$L_RECT  = 00000029
MNR_HDR$O_CLASSBITS = 00000073
MNR_HDR$O_REVOCLSBITS = 00000019
MNR_HDR$Q_BEGINNING = 00000005
MNR_HDR$Q_ENDING   = 0000000D
MNR_HDR$S_BEGINNING = 00000008
MNR_HDR$S_CLASSBITS = 00000010
MNR_HDR$S_COMMENT  = 0000003C
MNR_HDR$S_ENDING   = 00000008
MNR_HDR$S_FILE_HDR = 00000103
MNR_HDR$S_FILLER   = 00000020
MNR_HDR$S_FLAGS    = 00000004
MNR_HDR$S_LEVEL    = 00000008
MNR_HDR$S_REVOCLSBITS = 00000010
MNR_HDR$S_REVLEVELS = 00000080
MNR_HDR$S_TYPE     = 00000008
MNR_HDR$T_COMMENT  = 00000035
MNR_HDR$T_LEVEL    = 0000002D
MNR_HDR$T_REVLEVELS = 00000083

```

X 03
X 03
X 03
G X 03
X 03

MONITOR
Symbol table

- VAX/VMS Performance Monitor Utility

D 10

16-SEP-1984 01:59:24
5-SEP-1984 02:01:24

VAX/VMS Macro V04-00
[MONITOR.SRC]MONITOR.MAR;1

Page 179
(103)

MNR_HDR\$V_FILLER	=	00000000		
MNR_HDR\$W_COMLEN	=	00000071		
MNR_HOM\$K_PSIZE	=	00000008		
MNR_HOM\$SL_ELTCT	=	00000000		
MNR_HOM\$SL_RESERVED	=	00000004		
MNR_HOM\$SS_HOM_CLASS_PRE	=	00000008		
MNR_PRO\$B_PRI	=	0000000A		
MNR_PRO\$K_DSIZE	=	0000003B		
MNR_PRO\$K_FSIZE	=	00000040		
MNR_PRO\$K_PSIZE	=	00000008		
MNR_PRO\$K_REV0DSIZE	=	00000033		
MNR_PRO\$K_REV1DSIZE	=	0000003B		
MNR_PRO\$SL_BIOCNT	=	0000002F		
MNR_PRO\$SL_CPUTIM	=	0000002B		
MNR_PRO\$SL_DIOCNT	=	00000023		
MNR_PRO\$SL_EFWM	=	00000037		
MNR_PRO\$SL_EPID	=	00000033		
MNR_PRO\$SL_IPID	=	00000000		
MNR_PRO\$SL_PAGEFLTS	=	00000027		
MNR_PRO\$SL_PCTINT	=	00000004		
MNR_PRO\$SL_PCTREC	=	00000000		
MNR_PRO\$SL_STS	=	0000001F		
MNR_PRO\$SL_UIC	=	00000004		
MNR_PRO\$O_LNAME	=	0000000B		
MNR_PRO\$SS_LNAME	=	00000010		
MNR_PRO\$SS_PROCESS_CLASS	=	0000003B		
MNR_PRO\$SS_PRO_CLASS_PRE	=	00000008		
MNR_PRO\$W_GPGCNT	=	0000001B		
MNR_PRO\$W_PPGCNT	=	0000001D		
MNR_PRO\$W_STATE	=	00000008		
MNR_SYISB_MPCPUS	=	0000000D		
MNR_SYISB_TYPE	=	00000000		
MNR_SYISK_BALSETMEM	=	0000001E		
MNR_SYISK_CPUTYPE	=	00000026		
MNR_SYISK_MPWHILIM	=	00000022		
MNR_SYISK_NODENAME	=	0000000E		
MNR_SYISK_SIZE	=	0000002A		
MNR_SYISL_BALSETMEM	=	0000001E		
MNR_SYISL_CPUTYPE	=	00000026		
MNR_SYISL_MPWHILIM	=	00000022		
MNR_SYISQ_BOOTTIME	=	00000003		
MNR_SYISS_BOOTTIME	=	00000008		
MNR_SYISS_FILLER	=	0000000E		
MNR_SYISS_FLAGS	=	00000002		
MNR_SYISS_NODENAME	=	00000010		
MNR_SYISS_SYS_INFO	=	0000002A		
MNR_SYISS_TYPE	=	00000008		
MNR_SYIST_NODENAME	=	0000000E		
MNR_SYISV_CLUSMEM	=	00000000		
MNR_SYISV_FILLER	=	00000002		
MNR_SYISV_RESERVED1	=	00000001		
MNR_SYISW_FLAGS	=	00000001		
MNR_SYISW_MAXPRCCT	=	0000000B		
MODES_CLSNO	*****		X	03
MODES_ICOUNT	*****		X	03
MON_ERR	00001EB6	RG		03
MOVE_BARS	00000AE7	R		03

MOVE_CHD	00001D49	R	03
MOVE_CLASS_QUALS	0000012B	RG	03
MOVE_ITEMS	000012A0	R	03
MOVE_TOP8	00001383	R	03
MPCHECK	0000046A	RG	03
MRB	= 00000000		
MRBSA_COMMENT	= 0000002C		
MRBSA_DISPLAY	= 00000020		
MRBSA_INPUT	= 0000001C		
MRBSA_RECORD	= 00000024		
MRBSA_SUMMARY	= 00000028		
MRBSB_INP_FILES	= 00000042		
MRBSK_SIZE	= 00000045		
MRBSL_FLUSH	= 00000014		
MRBSL_INTERVAL	= 00000010		
MRBSL_VIEWING_TIME	= 00000018		
MRBSM_ALL_CLASS	= 00000400		
MRBSM_BY_NODE	= 00001000		
MRBSM_DISPLAY	= 00000001		
MRBSM_DISP_TO_FILE	= 00000020		
MRBSM_DIS_CL_REQ	= 00000100		
MRBSM_INDEFEND	= 00000010		
MRBSM_INP_CL_REQ	= 00000040		
MRBSM_MFSUM	= 00000800		
MRBSM_PLAYBACK	= 00000008		
MRBSM_PROC_REQ	= 00004000		
MRBSM_RECORD	= 00000002		
MRBSM_REC_CL_REQ	= 00000080		
MRBSM_SUMMARY	= 00000004		
MRBSM_SUM_CL_REQ	= 00000200		
MRBSM_SYSCLS	= 00002000		
MRBSQ_CLASSBITS	= 00000032		
MRBSQ_BEGINNING	= 00000000		
MRBSQ_ENDING	= 00000008		
MRBSS_BEGINNING	= 00000008		
MRBSS_CLASSBITS	= 00000010		
MRBSS_ENDING	= 00000008		
MRBSS_FLAGS	= 00000002		
MRBSS_MRB	= 00000045		
MRBSV_ALL_CLASS	= 0000000A		
MRBSV_BY_NODE	= 0000000C		
MRBSV_DISPLAY	= 00000000		
MRBSV_DISP_TO_FILE	= 00000005		
MRBSV_DIS_CL_REQ	= 00000008		
MRBSV_FILTER	= 0000000F		
MRBSV_INDEFEND	= 00000004		
MRBSV_INP_CL_REQ	= 00000006		
MRBSV_MFSUM	= 0000000B		
MRBSV_PLAYBACK	= 00000003		
MRBSV_PROC_REQ	= 0000000E		
MRBSV_RECORD	= 00000001		
MRBSV_REC_CL_REQ	= 00000007		
MRBSV_SUMMARY	= 00000002		
MRBSV_SUM_CL_REQ	= 00000009		
MRBSV_SYSCLS	= 0000000D		
MRBSW_CLASSCT	= 00000030		
MRBSW_FLAGS	= 00000043		

MONITOR
Symbol table

E 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 180
(103)

MRBPTR	*****	X	03	QUALSA_ALL	=	00000064	
MWAITLIST	*****	X	03	QUALSA_AVE	=	00000074	
NAMESTR	000022C0	RG	01	QUALSA-BEG	=	00000004	
NAME_COL	00000000	RG	01	QUALSA-BY NODE	=	00000054	
NAME_COL_BAR	= 00000002	G		QUALSA-CLASS	=	0000005C	
NAME_COL_MFSUM	= 00000001	G		QUALSA-COMM	=	0000004C	
NAME_COL_TAB	= 00000005	G		QUALSA-CPU	=	000000AC	
NEWXPOS	00000089	R	01	QUALSA-CUR	=	0000006C	
NO	= 00000000	G		QUALSA-DISP	=	00000034	
NORMAL	*****	X	03	QUALSA-END	=	0000000C	
NUMB ONLY	*****	X	03	QUALSA-FLUSH	=	0000001C	
OTHER VID	= 00000003			QUALSA-INP	=	0000002C	
OUTDSC	00001A03	RG	01	QUALSA-INT	=	00000014	
OUTDSC SIZE	= 000006A4			QUALSA-ITEM	=	000000BC	
PCBSV_RES	= 00000000			QUALSA-MAX	=	00000084	
PCENT_STR	0000256C	RG	01	QUALSA-MIN	=	0000007C	
PCTEN	00000E6F	R	03	QUALSA-PCENT	=	000000B4	
PC_BUFS	= 00000004	G		QUALSA-REC	=	0000003C	
PERFTABLE	*****	X	03	QUALSA-SUMM	=	00000044	
PLAY_STR	000023C4	R	01	QUALSA-TOPB	=	0000009C	
PREV_PD	000000E4	R	01	QUALSA-TOPC	=	0000008C	
PRINT_SCREEN	0000168D	R	03	QUALSA-TOPD	=	00000094	
PROCDISPS	= 00000005			QUALSA-TOPF	=	000000A4	
PROCESS_CLASS	= 00C00000			QUALSA-VIEW	=	00000024	
PROCHEAD_STR	00002403	RG	01	QUALSL_ALL	=	00000060	
PROCS_CLSNO	*****	X	03	QUALSL_AVE	=	00000070	
PROCS_PER_REC	00000016	R	01	QUALSL-BEG	=	00000000	
PROC_CINE	= 0000004B			QUALSL-BY NODE	=	00000050	
PROC_NRES_STR	00002676	R	01	QUALSL-CLASS	=	00000058	
PROC_RES_STR	00002631	R	01	QUALSL-COMM	=	00000048	
PROC_SETUP_STR	00002616	R	01	QUALSL-CPU	=	000000A8	
PROC_WRI_BUF	0000001A	R	01	QUALSL-CUR	=	00000068	
PROMPT_STR	000000E8	RG	01	QUALSL-DISP	=	00000030	
PRO_CLASS_PRE	= 00000000			QUALSL-END	=	00000008	
PTS_ESCB	00001BC5	R	03	QUALSL-FLUSH	=	00000018	
PTS_ESCF	00001C28	R	03	QUALSL-INP	=	00000028	
PTS_ESCG	00001C35	R	03	QUALSL-INT	=	00000010	
PTS_ESCH	00001C1B	R	03	QUALSL-ITEM	=	000000B8	
PTS_ESCJ	00001C12	R	03	QUALSL-MAX	=	00000080	
PTS_ESCK	00001C09	R	03	QUALSL-MIN	=	00000078	
PTS_ESCL	00001BCE	R	03	QUALSL-PCENT	=	000000B0	
PTS_ESCR	00001BDE	R	03	QUALSL-REC	=	00000038	
PTS_ESCU	00001BE7	R	03	QUALSL-SUMM	=	00000040	
PTS_ESCY	00001BEE	R	03	QUALSL-TOPB	=	00000098	
PTS_RET	00001C73	R	03	QUALSL-TOPC	=	00000088	
PTS_STAT	0000008D	R	01	QUALSL-TOPD	=	00000090	
PUTMSGSIZE	= 0000001A			QUALSL-TOPF	=	000000A0	
PUTMSGVEC	00002741	R	01	QUALSL-VIEW	=	00000020	
PUTSCRARG	0000271B	R	01	QUALSS-QUALIFIER_DESC	=	000000C0	
PUTS_ALTSET	= 00000000			QUALIFIER_DESC	=	00000000	
PUTS_REGSET	= 00000000			READ INPUT	*****	X	03
PUT_ESC_SEQ	00001C7B	R	03	REC_STR	000023E5	R	01
PUT_TO_SCREEN	00001B5D	RG	03	REG	000012C1	R	03
QEZ_RET	00000469	R	03	REGSET	*****	X	03
QLQ_RET	00000452	R	03	REG_BUFS	= 00000004	G	
QUAD_EQ_0	00000453	RG	03	REG-PROC	= 00000000		
QUAD_LT_QUAD	00000436	RG	03	REG_SET	00002711	R	01

MONITOR
Symbol table

F 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 181
(103)

RPBSL_BOOTR5	=	00000030			SS\$_BADATTRIB	=	00000034	G
RPBSV_MPM	=	0000000B			SS\$_BADCHKSUM	=	00000808	G
RSNS_MAX	=	0000000F			SS\$_BADESCAPE	=	0000003C	G
RWAITLIST		*****	X	03	SS\$_BADFILEHDR	=	00000810	G
SCANBUF		00001B63	R	03	SS\$_BADFILENAME	=	00000818	G
SCBSB_FLAGS	=	00000002			SS\$_BADFILEVER	=	00000820	G
SCBSK_SIZE	=	00000003			SS\$_BADIMGHDR	=	00000044	G
SCBSS_FILLER	=	00000006			SS\$_BADIRECTORY	=	00000828	G
SCBSS_FLAGS	=	00000001			SS\$_BADISD	=	00002004	G
SCBSS_STATS_BLOCK	=	00000003			SS\$_BADPARAM	=	00000014	G
SCBSV_ACTIVE	=	00000001			SS\$_BADQFILE	=	000003BC	G
SCBSV_CURRENT	=	00000000			SS\$_BADQUEUEHDR	=	00000394	G
SCBSV_FILLER	=	00000002			SS\$_BADRCT	=	0000216C	G
SCBSW_DBIDX	=	00000000			SS\$_BADSTACK	=	000002B4	G
SCHSC_MWAIT	=	00000002			SS\$_BADVEC	=	00002064	G
SCR\$B_DEVTYPE	=	00000008			SS\$_BEGOFFILE	=	00000938	G
SCR\$ERASE_LINE		*****	X	03	SS\$_BEGOFTAPE	=	00000A10	G
SCR\$ERASE_PAGE		*****	X	03	SS\$_BLOCKCNTERR	=	00000940	G
SCR\$SL_FLAGS	=	00000000			SS\$_BREAK	=	00000414	G
SCR\$PUT_LINE		*****	X	03	SS\$_BUFBYTALI	=	0000030C	G
SCR\$PUT_SCREEN		*****	X	03	SS\$_BUFFEROVF	=	00000601	G
SCR\$SCREEN_INFO		*****	X	03	SS\$_BUFNOTALIGN	=	00000324	G
SCR\$SET_CURSOR		*****	X	03	SS\$_BUGCHECK	=	000002A4	G
SCR\$SET_OUTPUT		*****	X	03	SS\$_CANCEL	=	00000830	G
SCR\$V_BOLD	=	00000000			SS\$_CANCELGRANT	=	00000E2A	G
SCR\$V_DECCRT	=	00000006			SS\$_CHAINW	=	00000C0B	G
SCR\$V_REVERSE	=	00000001			SS\$_CHANINTLK	=	0000004C	G
SCR\$V_SCREEN	=	00000000			SS\$_CLEARED	=	00002104	G
SCR\$V_UNDERLINE	=	00000003			SS\$_CLIFRCXT	=	00000980	G
SCRDSC		000020AF	RG	01	SS\$_CMODSUPR	=	0000041C	G
SCRDSC_SIZE	=	00000200			SS\$_CMODUSER	=	00000424	G
SCS_DISPNAME		00002166	RG	03	SS\$_COMMHARD	=	000020C4	G
SCS_FA0		000025E6	RG	01	SS\$_COMPAT	=	0000042C	G
SELECT_REV		00001CDC	R	03	SS\$_CONCEALED	=	00000691	G
SELECT_REV_LEVS		00001C98	RG	03	SS\$_CONNECFAIL	=	000020DC	G
SELECT_SET		00001C40	R	03	SS\$_CONTINUE	=	00000001	G
SHRS_OPENIN	=	00001098			SS\$_CONTROLC	=	00000651	G
SI	=	0000000F			SS\$_CONTROLO	=	00000609	G
SIGNALED_ERR		00001F0D	RG	03	SS\$_CONTROLY	=	00000611	G
SIGNAL_MON_ERR		00001F54	RG	03	SS\$_CREATED	=	00000619	G
SKIP_TO_CLASS		*****	X	03	SS\$_CTRLERR	=	00000054	G
SORT_PROCS		00001346	R	03	SS\$_CVTUNGRANT	=	0000213C	G
SPTR		*****	X	03	SS\$_DATACHECK	=	0000005C	G
SR_RSB		00001D48	R	03	SS\$_DATALETE	=	00002274	G
SS\$_ABORT	=	0000002C	G		SS\$_DATAOVERUN	=	00000838	G
SS\$_ACCONFLICT	=	00000800	G		SS\$_DBGEVENT	=	000006C1	G
SS\$_ACCVIO	=	0000000C	G		SS\$_DBGOPCREQ	=	000006A1	G
SS\$_ACEEXISTS	=	00000E42	G		SS\$_DEADLOCK	=	00000E0A	G
SS\$_ACEIDMATCH	=	000006B9	G		SS\$_DEBUG	=	0000046C	G
SS\$_ACLEMPY	=	000009D0	G		SS\$_DECOVF	=	000004A4	G
SS\$_ACLFULL	=	000009F8	G		SS\$_DEVACTIVE	=	000002C4	G
SS\$_ACPVAFUL	=	000002FC	G		SS\$_DEVALLOC	=	00000840	G
SS\$_ALRDYCLOSD	=	000006A9	G		SS\$_DEVALRALLOC	=	00000641	G
SS\$_ARTRES	=	00000474	G		SS\$_DEVASSIGN	=	00000848	G
SS\$_ASTFLT	=	0000040C	G		SS\$_DEVCMERR	=	0000032C	G
SS\$_BADACL	=	00000E3A	G		SS\$_DEVFOREIGN	=	00000064	G
SS\$_BADACLCTX	=	000021CC	G		SS\$_DEVICEFULL	=	00000850	G

MONITOR
Symbol table

G 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 182
(103)

SS\$DEVINACT = 000020D4 G
SS\$DEVMOUNT = 0000006C G
SS\$DEVNOTALLOC = 00000858 G
SS\$DEVNOTDISM = 000021B4 G
SS\$DEVNOTMBX = 00000074 G
SS\$DEVNOTMOUNT = 0000007C G
SS\$DEVOFFLINE = 00000084 G
SS\$DEVREQERR = 00000334 G
SS\$DGQINCOMP = 000009C0 G
SS\$DIRALLOC = 000009C8 G
SS\$DIRFULL = 00000860 G
SS\$DIRNOTEMPTY = 00002174 G
SS\$DISCONNECT = 0000204C G
SS\$DRVERR = 0000008C G
SS\$DUPDSKQUOTA = 000003DC G
SS\$DUPFILENAME = 00000868 G
SS\$DUPIDENT = 0000222C G
SS\$DUPLNAM = 00000094 G
SS\$DUPUNIT = 000021C4 G
SS\$ENDOFFILE = 00000870 G
SS\$ENDOF TAPE = 00000878 G
SS\$ENDOFUSRLBL = 00000970 G
SS\$ENDOF VOLUME = 000009A0 G
SS\$EOTIN = 00000C03 G
SS\$EXASTLM = 00002A04 G
SS\$EXBIOLM = 00002A0C G
SS\$EXBYTLM = 00002A14 G
SS\$EXCPUTIM = 000020AC G
SS\$EXDEPTH = 00000E1A G
SS\$EXDIOLM = 00002A1C G
SS\$EXDISKQUOTA = 000003EC G
SS\$EXENQLM = 00002A44 G
SS\$EXFILLM = 00002A24 G
SS\$EXGBLPAGFIL = 00002164 G
SS\$EXLNMQUOTA = 0000224C G
SS\$EXPGFLQUOTA = 00002A2C G
SS\$EXPORTQUOTA = 000003AC G
SS\$EXPRCLM = 00002A34 G
SS\$EXQUOTA = 0000001C G
SS\$EXQUOTAEND = 00002AFF G
SS\$EXQUOTASTRT = 00002A00 G
SS\$EXTIDXFILE = 00000880 G
SS\$EXTQELM = 00002A3C G
SS\$FCPREADERR = 00000888 G
SS\$FCPREPSTN = 00000988 G
SS\$FCPREWUNDER = 00000890 G
SS\$FCPSPACERR = 00000898 G
SS\$FCPWITERR = 000008A0 G
SS\$FILACCERR = 0000009C G
SS\$FILALRACC = 000000A4 G
SS\$FILELOCKED = 000008A8 G
SS\$FILENUMCHK = 000008B0 G
SS\$FILEPURGED = 00000679 G
SS\$FILESEQCHK = 000008B8 G
SS\$FILESTRUCT = 000008C0 G
SS\$FILNOTACC = 000000AC G
SS\$FILNOTCNTG = 000002AC G

SS\$FILNOTEXP = 000000B4 G
SS\$FLTDIV = 00000494 G
SS\$FLTDIV_F = 000004BC G
SS\$FLTQVF = 0000048C G
SS\$FLTQVF_F = 000004B4 G
SS\$FLTUND = 0000049C G
SS\$FLTUND_F = 000004C4 G
SS\$FORCEDERROR = 00002144 G
SS\$FORCEDEXIT = 0000217C G
SS\$FORMAT = 000000BC G
SS\$GPTFULL = 000000C4 G
SS\$GSDFULL = 000000CC G
SS\$HANGUP = 000002CC G
SS\$HEADERFULL = 000008C8 G
SS\$IDMISMATCH = 000003F4 G
SS\$IDXFILFULL = 000008D0 G
SS\$ILLBLKNUM = 000000DC G
SS\$ILLCDTST = 00002154 G
SS\$ILLCNTRFUNC = 000000E4 G
SS\$ILLEFC = 000000EC G
SS\$ILLIOFUNC = 000000F4 G
SS\$ILLBLAST = 00000968 G
SS\$ILLPAGCNT = 000000FC G
SS\$ILLSELF = 0000214C G
SS\$ILLSEQOP = 000002DC G
SS\$ILLSER = 00000104 G
SS\$ILLUSRLBLRD = 00000958 G
SS\$ILLUSRLBLWT = 00000960 G
SS\$INCOMPAT = 00000699 G
SS\$INCSEGTRA = 00002234 G
SS\$INCSHAMEM = 000022CC G
SS\$INCVOLLABEL = 0000010C G
SS\$INHCHME = 000004D4 G
SS\$INHCHMK = 000004CC G
SS\$INSFARG = 00000114 G
SS\$INSFBUFD = 0000033C G
SS\$INSFCDT = 000021AC G
SS\$INSFMAREG = 00000344 G
SS\$INSFMEM = 00000124 G
SS\$INSFRAME = 0000012C G
SS\$INSFSPTS = 00002044 G
SS\$INSFWSL = 0000011C G
SS\$INSSWAPSPACE = 00002264 G
SS\$INTDIV = 00000484 G
SS\$INTERLOCK = 0000038C G
SS\$INTQVF = 0000047C G
SS\$INVEXHLIST = 000022B4 G
SS\$INVLOGIN = 0000209C G
SS\$INVSECLASS = 000022C4 G
SS\$IVACL = 000021E4 G
SS\$IVADDR = 00000134 G
SS\$IVBUFLN = 0000034C G
SS\$IVCHAN = 0000013C G
SS\$IVCHAR = 000020CC G
SS\$IVCHNLSEC = 0000026C G
SS\$IVDEVNAM = 00000144 G
SS\$IVGSDNAM = 0000014C G

MONITOR
Symbol table

H 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 183
(103)

SS\$ _IVIDENT = 00002224 G
SS\$ _IVLOCKID = 00002124 G
SS\$ _IVLOGNAM = 00000154 G
SS\$ _IVLOGTAB = 0000015C G
SS\$ _IVLVEC = 0000203C G
SS\$ _IVMODE = 00000354 G
SS\$ _IVPROTECT = 000002F4 G
SS\$ _IVQUOTAL = 00000164 G
SS\$ _IVSECFLG = 0000016C G
SS\$ _IVSECIDCTL = 000002E4 G
SS\$ _IVSSRQ = 00000174 G
SS\$ _IVSTSFLG = 0000017C G
SS\$ _IVTIME = 00000184 G
SS\$ _JBCERROR = 0000218C G
SS\$ _KERNELINV = 00002244 G
SS\$ _LCKPAGFUL = 000000D4 G
SS\$ _LENVIO = 0000018C G
SS\$ _LINEABRT = 00000E02 G
SS\$ _LINKABORT = 000020E4 G
SS\$ _LINKDISCON = 000020EC G
SS\$ _LINKEEXIT = 000020F4 G
SS\$ _LKWSETFUL = 00000194 G
SS\$ _LNMCREATED = 000006B1 G
SS\$ _MBFULL = 000008D8 G
SS\$ _MBTOOSML = 0000019C G
SS\$ _MCHECK = 000002BC G
SS\$ _MCNOTVALID = 0000035C G
SS\$ _MEDOFL = 000001A4 G
SS\$ _MSGNOTFND = 00000621 G
SS\$ _MTLBLLONG = 00000304 G
SS\$ _MULTRMS = 0000202C G
SS\$ _MUSTCLOSEFL = 00000948 G
SS\$ _NOACLSUPPORT = 000022BC G
SS\$ _NOACNT = 0000284C G
SS\$ _NOALLSPOOL = 00002824 G
SS\$ _NOALTPRI = 0000286C G
SS\$ _NOAQB = 00000314 G
SS\$ _NOBUGCHK = 000028BC G
SS\$ _NOBYPASS = 000028EC G
SS\$ _NOCMEEXEC = 0000280C G
SS\$ _NOCMKRNL = 00002804 G
SS\$ _NODATA = 000001AC G
SS\$ _NODELEAVE = 0000223C G
SS\$ _NODETACH = 0000282C G
SS\$ _NODEVAVL = 000009B0 G
SS\$ _NODIAGNOSE = 00002834 G
SS\$ _NODISKQUOTA = 000003E4 G
SS\$ _NODOWNGRADE = 0000290C G
SS\$ _NOENTRY = 000009D8 G
SS\$ _NOEXQUOTA = 0000289C G
SS\$ _NOFILACC = 000022AC G
SS\$ _NOGROUP = 00002844 G
SS\$ _NOGRPNAM = 0000281C G
SS\$ _NOGRPPRV = 0000291C G
SS\$ _NOHANDLER = 000008F8 G
SS\$ _NOHOMEBLK = 000008E0 G
SS\$ _NOIOCHAN = 000001B4 G

SS\$ _NOLCLMEDA = 000021F4 G
SS\$ _NOLICENSE = 00002194 G
SS\$ _NOLINKS = 0000027C G
SS\$ _NOLISTENER = 0000215C G
SS\$ _NOLOCKID = 00000E12 G
SS\$ _NOLOGNAM = 000001BC G
SS\$ _NOLOGTAB = 00002294 G
SS\$ _NOLOG_IO = 0000283C G
SS\$ _NOMBX = 00000274 G
SS\$ _NOMOREACE = 000009E0 G
SS\$ _NOMOREFILES = 00000930 G
SS\$ _NOMORELOCK = 00000A08 G
SS\$ _NOMORENODE = 00000A00 G
SS\$ _NOMOREPROC = 000009A8 G
SS\$ _NOMOUNT = 0000288C G
SS\$ _NONETMBX = 000028A4 G
SS\$ _NONEXDRV = 000001C4 G
SS\$ _NONEEXPR = 000008E8 G
SS\$ _NONLOCAL = 000008F0 G
SS\$ _NOOPER = 00002894 G
SS\$ _NOP1VA = 00002024 G
SS\$ _NOPFNMAP = 000028D4 G
SS\$ _NOPHY_IO = 000028B4 G
SS\$ _NOPRI = 00000024 G
SS\$ _NOPRIVEND = 000029FF G
SS\$ _NOPRIVSTRT = 00002800 G
SS\$ _NOPRMCEB = 00002854 G
SS\$ _NOPRMGBL = 000028C4 G
SS\$ _NOPRMJNL = 00002904 G
SS\$ _NOPRMMBX = 0000285C G
SS\$ _NOPSWAPM = 00002864 G
SS\$ _NOQFILE = 000003C4 G
SS\$ _NOREADALL = 00002924 G
SS\$ _NOREGAVIL = 000021FC G
SS\$ _NOREGSUIT = 00002204 G
SS\$ _NORMAL = 00000001 G
SS\$ _NOSECURITY = 00002934 G
SS\$ _NOSETPRV = 00002874 G
SS\$ _NOSHARE = 0000292C G
SS\$ _NOSHMBLOCK = 000003B4 G
SS\$ _NOSHMEM = 000028DC G
SS\$ _NOSHRIMG = 000021BC G
SS\$ _NOSIGNAL = 00000900 G
SS\$ _NOSLOT = 0000039C G
SS\$ _NOSOLICIT = 00000284 G
SS\$ _NOSUCHDEV = 00000908 G
SS\$ _NOSUCHFILE = 00000910 G
SS\$ _NOSUCHID = 000021EC G
SS\$ _NOSUCHNODE = 0000028C G
SS\$ _NOSUCHOBJ = 000020A4 G
SS\$ _NOSUCHPGM = 0000220C G
SS\$ _NOSUCHSEC = 00000978 G
SS\$ _NOSUCHUSER = 00002084 G
SS\$ _NOSYSGBL = 000028CC G
SS\$ _NOSYSLCK = 000028F4 G
SS\$ _NOSYSNAM = 00002814 G
SS\$ _NOSYSPRV = 000028E4 G

MONITOR
Symbol table

I 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 184
(103)

SS\$_NOTALLPRIV	= 00000681	G
SS\$_NOTAPEOP	= 00000264	G
SS\$_NOTCREATOR	= 00000384	G
SS\$_NOTFILEDEV	= 000001CC	G
SS\$_NOTINSTALL	= 00002014	G
SS\$_NOTINTBLSZ	= 000001D4	G
SS\$_NOTLABELMT	= 000001DC	G
SS\$_NOTMODIFIED	= 00000659	G
SS\$_NOTMPJNL	= 000028FC	G
SS\$_NOTMPMBX	= 0000287C	G
SS\$_NOTNETDEV	= 000002EC	G
SS\$_NOTPRINTED	= 00002184	G
SS\$_NOTQUEUED	= 000009B8	G
SS\$_NOTRAN	= 00000629	G
SS\$_NOTSQDEV	= 000001E4	G
SS\$_NOTVOLSET	= 00000998	G
SS\$_NOUPGRADE	= 00002914	G
SS\$_NOVOLACC	= 000022A4	G
SS\$_NOVOLPRO	= 000028AC	G
SS\$_NOWORLD	= 00002884	G
SS\$_NOWRT	= 000003FC	G
SS\$_OPCCUS	= 00000434	G
SS\$_OPCDEC	= 0000043C	G
SS\$_OPINCOMPL	= 000002D4	G
SS\$_OPRABORT	= 000020B4	G
SS\$_OVRDSKQUOTA	= 00000669	G
SS\$_OVRMAXARG	= 0000227C	G
SS\$_PAGOWNVIO	= 000001EC	G
SS\$_PAGRDERR	= 00000444	G
SS\$_PARENT_DEL	= 00002254	G
SS\$_PARITY	= 000001F4	G
SS\$_PARNOTGRANT	= 00002134	G
SS\$_PARNOTSYS	= 0000225C	G
SS\$_PARTESCAPE	= 000001FC	G
SS\$_PARTMAPPED	= 00000E22	G
SS\$_PATHLOST	= 000020FC	G
SS\$_PFMBSY	= 00000204	G
SS\$_PGMLDFAIL	= 00002214	G
SS\$_PGMSTDALN	= 0000221C	G
SS\$_PLHLDR	= 00000404	G
SS\$_POWERFAIL	= 00000364	G
SS\$_PRIVINSTALL	= 00002054	G
SS\$_PROTINSTALL	= 0000205C	G
SS\$_PROTOCOL	= 00002074	G
SS\$_PSTFULL	= 0000020C	G
SS\$_QFACTIVE	= 000003CC	G
SS\$_QFNOTACT	= 000003D4	G
SS\$_RADRMOD	= 0000044C	G
SS\$_RDDELDATA	= 00000661	G
SS\$_REJECT	= 00000294	G
SS\$_RELINK	= 0000200C	G
SS\$_REMOTE	= 00000649	G
SS\$_REMRSRC	= 0000206C	G
SS\$_RESET	= 0000210C	G
SS\$_RESIGNAL	= 00000918	G
SS\$_RESULTOVF	= 00000214	G
SS\$_RETRY	= 00000E32	G

SS\$_RIGHTSFULL	= 000009E8	G
SS\$_ROPRAND	= 00000454	G
SS\$_SECTBLFUL	= 0000021C	G
SS\$_SERIOUSEXCP	= 000021D4	G
SS\$_SHACHASTA	= 00002284	G
SS\$_SHACPYINP	= 000022D4	G
SS\$_SHARTOOBIG	= 0000201C	G
SS\$_SHMGSNOTMAP	= 0000036C	G
SS\$_SHMNOTCNCT	= 0000037C	G
SS\$_SHRIDMISMAT	= 000020BC	G
SS\$_SHUT	= 0000208C	G
SS\$_SSFAIL	= 0000045C	G
SS\$_SUBLOCKS	= 0000212C	G
SS\$_SUBRNG	= 000004AC	G
SS\$_SUPERSEDE	= 00000631	G
SS\$_SUSPENDED	= 000003A4	G
SS\$_SYNCH	= 00000689	G
SS\$_SYSAPMAX	= 00007FFF	G
SS\$_SYSAPMIN	= 00007E00	G
SS\$_SYSVERDIF	= 00000671	G
SS\$_TAPEPOSLOST	= 00000224	G
SS\$_TBIT	= 00000464	G
SS\$_TEMPLATEDEV	= 000021DC	G
SS\$_TERMNETDEV	= 0000228C	G
SS\$_THIRDPARTY	= 0000207C	G
SS\$_TIMEOUT	= 0000022C	G
SS\$_TOOMANYLNAM	= 00000374	G
SS\$_TOOMANYREDS	= 0000211C	G
SS\$_TOOMANYVER	= 00000990	G
SS\$_TOOMUCHDATA	= 0000029C	G
SS\$_UNASEFC	= 00000234	G
SS\$_UNREACHABLE	= 00002094	G
SS\$_UNSAFE	= 0000023C	G
SS\$_UNSLICIT	= 00002114	G
SS\$_UNWIND	= 00000920	G
SS\$_UNWINDING	= 00000928	G
SS\$_VALNOTVALID	= 000009F0	G
SS\$_VASFULL	= 00000244	G
SS\$_VCBROKEN	= 0000219C	G
SS\$_VCCLOSED	= 000021A4	G
SS\$_VECFULL	= 00002034	G
SS\$_VECIUSE	= 0000024C	G
SS\$_VOLINV	= 00000254	G
SS\$_VOLERR	= 0000226C	G
SS\$_WAITUSRLBL	= 00000950	G
SS\$_WASCLR	= 00000001	G
SS\$_WASECC	= 00000639	G
SS\$_WASECT	= 00000009	G
SS\$_WRITLCK	= 0000025C	G
SS\$_WRONGACP	= 0000031C	G
SS\$_WRONGNAME	= 0000229C	G
STACK TOP	000011C1	R 03
STARTPOS	= 00000032	
START XPOS	= 00000026	
STATELIST	*****	X 03
STATES CLSNO	*****	X 03
STATHEAD_STR	0000251E	RG 01

MONITOR
Symbol table

J 10
- VAX/VMS Performance Monitor Utility

16-SEP-1984 01:59:24 VAX/VMS Macro V04-00
5-SEP-1984 02:01:24 [MONITOR.SRC]MONITOR.MAR;1

Page 185
(103)

STATS = 00000005
STATS_BLOCK = 00000000
STATUS_PARMs 000023F7 RG 01
STATUS_STR 000022FF RG 01
STAT_HEAD 00002570 RG 01
STAT_LONG 0000257C RG 01
STSSR_ERROR = 00000002
STSSS_FAC_NO = 0000000C
STSSV_FAC_NO = 00000010
SUMMARY_INIT 00000A5B RG 03
SUMMARY_TOP 0000145D RG 03
SUMMLINE_STR 00002371 RG 01
SUMM_STR 000023D3 R 01
SYIS_CPU = 00002000
SYIS_NODENAME = 000010D9
SYSSASCTIM ***** GX 03
SYSSASSIGN ***** GX 03
SYSSCMKRNL ***** GX 03
SYSSFAOL ***** GX 03
SYSSGETCHN ***** GX 03
SYSSGETSYIW ***** GX 03
SYSSGETTIM ***** GX 03
SYSSPUTMSG ***** GX 03
SYSSQIOW ***** GX 03
SYSSSETIMR ***** GX 03
SYSSSTRNLOG ***** GX 03
SYSSWAITFR ***** GX 03
SYSCMD_DESC 000000B1 R 01
SYSNOD_NAM 000000C4 R 01
SYSOUT_TYPE 00002615 R 01
SYSTEMS FACILITY = 00000000 G
SYSTEM_CLSNO ***** X 03
SYS_BOX_STR_ADDR 00000085 RG 01
SYS_BOX_STR_G ***** X 03
SYS_BOX_STR_H ***** X 03
SYS_BOX_STR_LEN 00000083 RG 01
SYS_BOX_STR_LEN_G ***** X 03
SYS_BOX_STR_LEN_H ***** X 03
SYS_DATA_ADDR 0000007B RG 01
SYS_DATA_LEN 0000007F RG 01
SYS_FAC_NO = 00000000
SYS_FAO_STR ***** X 03
SYS_HEAD_STR 000024BD RG 01
SYS_INFO = 00000000
SYS_SUMMLINE_STR 0000239E RG 01
SYS_TIME_STR 00002362 RG 01
SYS_TOP_VEC 0000003B RG 01
S_TOP_TICKS 00000037 R 01
S_TOP_TIME 0000002F R 01
TABHEAD_STR 00002479 RG 01
TAB_LWORDS = 00000008
TEMP_4_BLOCK = 00000000
TIME_STR 00002353 RG 01
TITLE_STR 00002315 RG 01
TM4SA_BUFFERS = 00000008
TM4SA_ITMSTR = 00000004
TM4SK_SIZE = 00000010

TM4SL_ECOUNT
TM4SL_FLTSECS
TM4SS_TEMP_4_BLOCK
TOPB
TOPBAR
TOPB_PROC
TOPC
TOPC_PROC
TOPD
TOPD_PROC
TOPF
TOPF_PROC
TOPLNO
TOPSTR
TOP_DIFFS
TOP_PROCS
TOP_TICKS
TOP_TIME
TRANSFORMS
TXT_DESC
TXT_LENGTH
TXT_START
UNKNOWN NODE
UPD_PC_MIN_MAX
VID_BAR
VIEWING DEL
VT100_ACTSET
VT100_CURSET
VT100_REGSET
VT55CWIDTH
VT55HEIGHT
VT55WIDTH
VT55XINCR
VT5X
VTDATALINES
VTHEIGHT
VTWIDTH
WPR_RET
WRITE_HEADER
WRITE_PROC RECORDS
WRITE_RECORD
YES

= 00000000
= 0000000C
= 00000010
000012CB R 03
= 000026EA R 01
= 00000003
000012C1 R 03
= 00000001
000012C6 R 03
= 00000002
000012D0 R 03
= 00000004
= 000026C6 R 01
000026C3 R 01
000013ED R 03
00000022 R 01
0000002B R 01
00000023 R 01
000003F0 R 03
0000272F R 01
0000272F R 01
00002733 R 01
000025F1 R 01
00000971 R 03
= 00000061
***** X 03
00002702 R 01
0000270D R 01
000026F6 R 01
= 0000004A G
= 000000EC
= 00000200
00C000D4 RG 01
= 00000001
= 0000000F G
= 00000018 G
= 00000050 G
00000586 R 03
***** X 03
0000048A RG 03
***** X 03
= 00000001 G

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
MONDATA	000027A9 (10153.)	01 (1.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC QUAD
\$ABSS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$MONCODE	00002175 (8565.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	48	00:00:00.13	00:00:01.28
Command processing	161	00:00:00.94	00:00:05.54
Pass 1	1015	00:00:41.65	00:01:30.83
Symbol table sort	0	00:00:05.61	00:00:08.40
Pass 2	446	00:00:16.25	00:00:40.35
Symbol table output	2	00:00:00.96	00:00:01.81
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1675	00:01:05.57	00:02:28.29

The working set limit was 3300 pages.
252219 bytes (493 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3159 non-local and 394 local symbols.
7063 source lines were read in Pass 1, producing 157 object records in Pass 2.
67 pages of virtual memory were used to define 55 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[MONITOR.OBJ]MONLIB.MLB;1	11
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	8
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	32
TOTALS (all libraries)	51

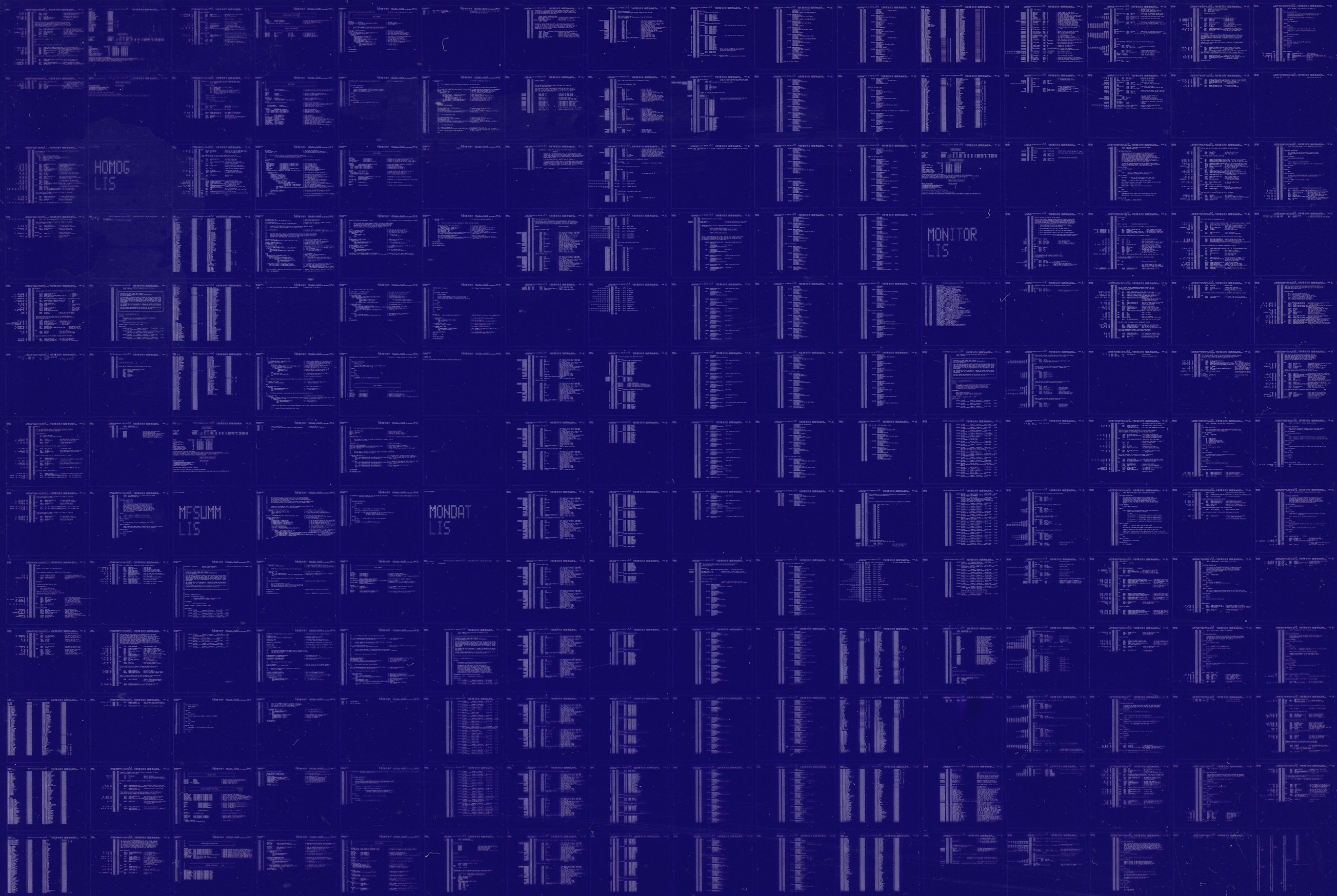
3099 GETS were required to define 51 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:MONITOR/OBJ=OBJ\$:MONITOR MSRC\$:MONITOR/UPDATE=(ENH\$:MONITOR)+EXECMLS/LIB+LIB\$:MONLIB/LIB

0240 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0241

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY